2017

Queer Practices, Queer Rhetoric, Queer Technologies: Studies of Digital Performativity in Gendered Network Culture

Gerald Jackson
University of South Carolina

Queer Practices, Queer Rhetoric, Queer Technologies: Studies of Digital Performativity in Gendered Network Culture

By

Gerald Jackson

Bachelor of Arts
Southern Illinois University Edwardsville, 2010

Master of Arts
Southern Illinois University Edwardsville, 2012

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

English

College of Arts and Sciences

University of South Carolina

2017

Accepted By:

Pat Gehrke, Major Professor

Holly Crocker, Committee Member

Kevin Brock, Committee Member

Jie Guo, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

Dedication

I dedicate my work to my family, Cindy and Lexi.

Acknowledgements

There is a long list of people without whom I would never have completed this work. Thanks to my committee for their work on this. Some of my best ideas and arguments were framed in courses taught by Holly Crocker, Pat Gehrke, and Jie Guo. Kevin Brock has also been a wonderful technical mind to bounce ideas off of, not to mention a source of relief when organizing D&D campaigns.

Most of all, I want to acknowledge my wife, Cindy. Without her, I would never have come this far. She has taught me more about myself than a decade in the university, and I look forward to shaping our own futures together.

Abstract

The question of gender and gender representation has been an issue for STEM fields like computer science and software engineering for decades. This dissertation argues that the impact of labor practices in such fields promotes gender disparity by masking gender, and often sexuality, behind myths of mastery and correctness. In this dissertation, I approach technical objects like computer code and protocol specifications from the BitTorrent and Bitcoin software packages, and argue that gendered forms of labor that have existed since the inception of computer programming as a profession are evident in technical documents like code. Furthermore, I argue that this labor is a communicative labor steeped in political rhetoric and cultural practices. As such, moving past a relative examination of code or software as a text, or as a procedure, invites a performative investigation of code. Drawing from theories in feminist technology and queer computational studies, I argue that gender, activism, and power relations are implicit in these technical objects, and that a rhetoric of gender and computation can emerge as a critical and practical practice in fields such as software engineering and technical communication.

Table of Contents

List of Figures

Chapter 1
Introduction

Digital Technology is not gender neutral. In terms of communicative technology, there is a sense in which scholars understand this intuitively. And yet we all, at some time, bracket the potential implications of gendered media in terms of functionality and correctness. This could be for a number of reasons, but first and foremost I posit that the seeming break between what serves as cultural technology and what serves as functional technology rests in how we ask questions regarding the more esoteric aspects of digital media. Code studies, an emerging body of scholarship in media studies, rhetorical studies, and feminist studies has emerged as a response to the notion of computer programming languages as a cultural form of expression. This seems perfectly reasonable in that code is, in many senses, a language. But it is precisely when we take in the implications of something like computer source code, and the understanding that code constructs an audience in a way that necessarily includes both humans and machines (Hayles 2010), which in turn limits the application of code as language. And yet, as scholars such as Annette Vee, James Brown Jr., and queer digital artist Zach Blas have argued, code is, and always is, a communicative function of cultural labor.

Following that understanding of a technological object like code, I put forth the central question of my argument: what is the communicative function of code? In exploring something like code, we typically understand it as we would understand a text, which means taking pieces of it and literally reading algorithms as pieces of meaning-

making. And yet code, like writing more broadly, is constituted not simply by the intention of an author creating software, but by an

immense body of labor, knowledge, and technology that frames any capacity for meaningful interaction. Even the simplest line of code has a host of attendant software and code libraries that make it meaningful both technologically and culturally, rooted in decades of research into mathematics, engineering, and philosophy. A given selection of code itself might be only a small portion of a larger program, a logic that spans millions of lines of code written by dozens of individuals. And, most importantly for my argument here, all that labor and all that writing of code is embedded in cultural understandings of what it means to work collaboratively and collectively through new techniques of development in a new medium. And these cultural assumptions have largely gathered in a constellation of notably masculine characteristics. The rise of the "brogrammer" culture in Silicon Valley, of women in bikinis as videogame conferences, all serve as a backdrop for a larger question of how masculinity is repurposed and defined within a technological context. And, further, as calls continue for an increased awareness and action regarding the representation of women in technical fields, how do scholars consider the operationalized nature of these masculine characteristics in the technology and professional practices they use every day?

The history of computer programming (which I will cover very briefly in chapter 2) has been shown by scholars as one structured on the "clerical" work of women. As men stepped into the role of "the programmer" with the advent of software as we know it, then a shift also occurred in how we understand technology as not only an object of use, but as a practice and, ultimately, as a cultural paradigm. What was once seen as an

ultimately clerical occupation became scientific and challenging, which led to the institutionalization of a gender disparity that recognized men as users of technology, and women simply as (and later, replaced by) technology. Wendy Chun (2004) writes that it is not simply the culture of technology that repels women from technological fields, but it is the implicit culture of masculinity within the technology itself that structures the very condition of work within those fields. That is, the history of computing is not a theater of gender representation on top of a neutral technological substrate, but rather a relationship between practices in technology and how they emerge from gendered history.

This is where I situate my argument. If a category like gender is such that it permeates even seemingly esoteric technology like computer code execution or networked communication technologies, where is the space in which a field like rhetoric investigates this phenomenon? I argue that initial inquiry must incorporate studies in technology alongside insight from scholars working with gender and technology in such a way as to recognize how the text of something like computer code (or other logical protocols, as I will discuss in chapters 3 and 4) actually functions computationally and culturally. I argue here that a significant point of entry will tie into programming techniques and practices that permeate small and large-scale software, which in turn links software development to a question of communication and agency.

Agency and Rhetoric

As a multi-issue debate over rhetorical agency and pedagogy unfolded over the pages of the *Rhetoric Society Quarterly* during 2004 and 2005, I became aware of how these terms structure rhetoric's fashioning of an ethical or civic imperative built around both

rhetoric's humanist tradition and its poststructural or posthuman trajectories. Humanist scholars such as Michael Leff and Cheryl Geisler take posthuman critique of subjectivity and agency within a frame of complex social and historical mediation that complicates agency, while situating rhetoric as realm of intentional language use, and deliberative political or civic action. Others such as Jenny Rice, Byron Hawk, and Colin Brooke attempt to discuss agency as a networked effect, of situatedness and the interplay of actors producing rhetoric effects in circulatory channels of meaning-making. In many cases, one can be adopted by the other, or assumptions of one assimilated within the framework of the other: Michael Leff (2012) adopts a more nuanced concept of agency via social and historical mediation that problematizes the singular author or rhetor while asserting the civic values from which that model arises. Similarly, Karen Lunsford (2004) argues that humanist rhetoric can take into account the critiques of posthumanism, but it must not give up its ethical and civic imperatives that stems from a humanist account of agency and subjectivity, arguing that some posthuman scholars see rhetorical study as a study of the "content" of rhetorical effect, as opposed to a Humanist tradition of civic/pedagogical instruction through a teachable *techne* of rhetoric that plays out in deliberative, normative political frameworks (such as "democracy"). However, what has become apparent to me and others is that the introduction of networked, digital technologies as a fundamental reality of communication brings into relief how these seemingly opposed concepts (posthuman critique and humanist *techne*) converge.

In "How Ought We to Understand the Concept of Rhetorical Agency? Report from the ARS," Cheryl Geisler (2004) summarizes and responds to the driving topic of the titular ARS meeting on the question of rhetorical agency and the "future" of the

discipline of rhetoric. Geisler writes that the question foreshadows “the complex interplay between rhetoric’s interpretive project and rhetoric’s educational mission” (9). In an accompanying piece in the same issue, Leff and Lunsford write in "Afterword: A Dialogue" that rhetoric's "birthright" is a focus on civic education rooted in rhetorical training. Lunsford, as one half of the titular dialogists, remarks that she believes rhetoric’s role in the university is “developing competent and ethical citizens” through rhetorical instructions, although some other theorists might view rhetoric as a “content” discipline focused on history and theory (Leff and Lunsford 7). Both Leff and Lunsford argue that rhetoric's claim to a pedagogical imperative is at odds with its imbrication with the "German model of the university" and thus a turn towards theory, research, and criticism (7). Geisler, for her part, acknowledges the moves to rethink agency but argues both that agency is a possession, and that keeping rhetorical theory within its classical, humanist tradition keeps rhetoric tied to a pedagogical mission grounded in political identity and change (Geisler 15). Both essays trace a similar map of the field, in that they argue that the lynchpin of both a rhetorician's agency and the agency of the field of rhetoric as a whole is tied to a theory of action and effect tied to an intentional agent (Leff and Lunsford 15). This suggests that the loss of a concept of agency (as they ascribe to a poststructural, postmodern, and/or posthumanist position) itself leaves rhetoric with little to do, precisely because the mission of rhetoric “as a productive art” (that it teach and describe efficacious action in the world, and that this art is teachable) no longer has a content or meaning within disciplinary or institutional contexts (and this she ascribes primarily to the split between an analytic rhetorical approach and its productive counterpart). Rhetoricians will have nothing to say about the “potential and obligations”

of students in rhetorical classrooms, and while functioning as critics rhetoricians will have no purchase from which to describe, articulate, or empower rhetorical agent (16). Rhetoric as a political and pedagogical concept loses its ability to be either beyond the critiques of poststructural or posthumanist thought.

In their response essay "'Ouija Board, Are There Any Communications?' Agency, Ontotheology, and the Death of the Humanist Subject, or, Continuing the ARS Conversation," Christian Lundberg and Joshua Gunn (2005) write,

> [a]lthough [Geisler] rightfully notes that the discussion question of rhetorical agency often melds the ontological (what?) with the ethical (how?), she and others nevertheless seek to infer the former by presenting the latter evangelically, stressing the need of moral activism for civic salvation and charging those with postructural and/or posthumanist sympathies as advocating a nihilistic brand of Calvinist passivity. (86)

The delineation of the subject or agent *a priori* to a discussion of political action implies, at its heart, an *a priori* field of political action, dissent, and, as a result, rhetoric grounded within that agent position. Leff and Lunsford share concern with Geisler that the loss of the "autonomous human agent" for rhetoric signifies a loss of rhetorical possibility and the capacity to act politically. Lundsford in particular writes that the field, in negotiating a way in which to recognize the contingency of rhetorical action without falling into the trap of the death of the enlightenment subject, can retain both rhetoric's commitment to human autonomy and political action (Leff and Lunsford 65). Leff, Lunsford, and Geisler all connect to the understanding that rhetoric, as an intentional *techne* deployed for political change and as an art that is teachable, requires some fashioning (however problematized) of the human subject/agent within discursive action before or

concurrently with a field of possibility, and this field is constituted as a civically/deliberatively engaged humanist subject. Geisler writes in her argument for agents and agency is twofold: an argument from fact ("facts on the ground do not appear to support the proposition that rhetorical agency is illusionary") and from value:

> If neither our students nor our fellow citizens have such potential [rhetorical and civic] obligations . . . we may sidestep these questions of potential and obligations . . . but only at the cost of the irrelevancy of rhetoric. (Geisler 16)

In all cases, Geisler, Lunsford, and Leff are engaged with rhetoric's mission as one of educating, civically and ethically, future citizens and civic subjects in a public world, and agency as observed must be a description of such a position.

From these comments, I highlight four claims specifying where the rub between critical/ theoretical rhetoric and the pedagogical imperative of rhetoric lies. I also respond to these claims in order to build from them and fashion a jumping-off point for further discussion:

(1) The framing of rhetoric as that which must function as an education for some arguably productive, ethical citizen subject asserts and *a priori* subject position which, in its definition, is deemed as ontologically necessary for agency (or the possession of agency) and thus defines itself circularly. For example, Geisler's assertion that rhetorical agency *exists*, because "subaltern groups . . . are not without agency altogether" (through discourse, performance, etc.), situates agency as a "something" already always figured within civically-oriented power relations rather than reading the poststructural/posthumanist critique of the agent as a critique of that very same normative, boundary-drawing move, of defining a frame of democratic and deliberative

discourse (or the struggles for such discursive opportunities) beforehand (15). This can also be seen in Lunsford's mobilization of feminist critiques of poststructural thought, which undermine the notion of a postructural, productive agent position by arguing that theory, divorced from productive concerns of rhetorical agency, supplants the very subjectivity and agency women have fought for for years. Lundberg and Gunn argue, however, that "it does not necessarily follow that agency must be rooted in an autonomous, intending human agent . . . Nor does the mere fact of movement allow for the discernment of the movement's cause without pre-reading the situation through the lens of a humanist account of an agent producing agency" (Lundberg and Gunn 92). Lundberg and Gunn point out, rightfully so, that the definition of the agent is a definition of horizons of potential actions, possible positions, and largely of the possibilities of rhetoric as a whole. More specifically, Lundberg and Gunn's critique of Geisler is that her reading of poststructural and posthumanist thought is not very generous: the critique itself is to ward off the dangers of deciding beforehand the conditions of agency across rhetoric as a practice. In this way, we can see the different opinions between those who see a split between the critical and productive aspects of rhetoric, and those who see the critical as responding to and problematizing, in this case, the productive.

(2) The assertion of a politics *a priori* from which rhetorical action finds its meaning. Ronald Walter Greene (2004), in many ways, addresses this concern in his essay "Rhetoric and Capitalism: Rhetorical Agency as Communicative Labor." Greene's critique is that it is a "commonplace to describe rhetorical agency as political action," and by default that political action will fall within civically-minded modes of persuasion and discourse (188). He writes that "rhetorical agency often takes on the characteristics of a

normative theory of citizenship" in which the "good citizen persuades and is persuaded by the gentle force of the better argument" (188). This is a rhetorical structure of orator-statesman that pre-supposes publics, campaigns, elections, speeches, and debates. Strategies and effectiveness are polarized within these concepts so that effectiveness becomes politically and ethically compromised as to prioritize discourse within these particular formations. Rhetorical agency, formulated with ostensibly republican-democratic modes of deliberative discourse, and rhetorical agency *qua* political change becomes *a priori* a mode of engagement within rational ("gentle") discursive engagement.

Greene notes that the recognition of the public political sphere, as a form of marketing and capital engagement that exists strictly within the logics of this political rhetoric, makes manifests a rhetoric of money, advertising, and spectacular messages with "effectiveness" modeled on one's ability to participate within these particular discourses. Therefore, while there are various strategies through which to think about how rhetoric functions persuasively in this civic context, these strategies do not address how rhetorical agency--as the ability to effect change within an ultimately ethically corrupt system--functions without being defined by said structures. This encourages rhetorical scholars to become "moral entrepreneurs scolding, correcting, and encouraging the body politic to improve the quality and quantity of political participation" (189). That is, rhetoric as a civic endeavor within a model of political communication stands either in support of, or in opposition to, normative power, and the pedagogical drive of rhetoric to teach "effective" rhetoric not only must acknowledge these structures, but in some ways, acquiesce to them (198). Thinking about the practices, rather than the content, of

rhetorical engagement in this context means that the discursive structures of engagement (debate, protest, public gatherings, letter writing, etc.) are at once the primary means of exercising rhetorical agency and at the same time the means adjudicated by a system not directly indebted to the recognition of novel rhetorical agency. Rhetoric is always suspended within a dialectic of structure and resistance.

Outside of Greene's focus on labor and communication, Erin Rand (2014), in *Reclaiming Queer,* writes that theorizing agency as oppositional always already limits the potential for that agency, that the true indeterminacy of agency only comes when we move from this dialectic. Drawing from Karolyn Kohr Campbell's (2005) "Agency, Promiscuous and Protean," Rand takes as a starting point Campbell's argument that agencies are "points of articulation" and that agency is "textual," arguing that agency only works through persuasive acts (texts, in Rand's case) that are recognizable or intelligible within particular contexts (7). While a democratic-republican context would recognize a particular set of practices as legitimate, getting caught within the dialectic would limit and over-determine the possibility of agencies to emerge. Rather, Rand posits a "queer" agency that represents the excess of rhetorical action, the inventiveness and potential for action to work. "Queer" agency, in her theories, does not presuppose oppositional qualities as this inscribes agential possibility within the conditions of opposition (that is, it is always already oppositional, which is to say, circumscribed within the logics of the oppositional discourse) (Rand 23). Rather, it posits that there are two crucial gaps within theorizations of queer agency: first, that the agent (in this case, a "queer" agent) does not contain the capacity to determine the effectiveness of its actions (just being a queer agent does not mean that one acts queerly); and next, that the

effectiveness of and action cannot be determined by that action, only by the "uptake of its effects (24). This requires, in many ways, indeterminate ideas of the agent.

(3) That communication is transparent and transcendent (Lundberg and Gunn 84). A small point, but with huge ramifications: that persuasion, discourse, rhetoric, exist unfettered or, if mediated, mediated as a contained unity or whole (my discourse is mutated through mediation precisely because there is a discrete "production/produced" relationship at the core of all of these points, the *techne* of rhetoric). This suggests a wholeness to communication, a separateness of discourse and its transmission or propagation. A small sentence within Lundberg and Gunn's critique links the figure of the Ouija board to a theory of communication as a connection, a conduit between two subjects through which a sharing of discourse occurs. Implicitly, like a game of Telephone, this implies that there is "a" communication happening in which success, failure, or even persuasive effects resides within the communication itself. In "Refitting Fantasy," Gunn writes that our notions of communication being something that connects transcendent subjects are based on "fantasy," the fantasy itself a mechanism of mediation that allows communicators to engage with behaviors and effects. That is, communication is never a direct link but an engagement with effects in the world (a "cold read," as Gunn uses the metaphor of a psychic reading). Both Gunn and Lundberg pick this back up in "Ouija Board" to point out how embedded a rhetorical construct of agency is within a "ontotheological framework" of agency, in that however problematized, it implies and harkens to the dissolution of the Cartesian self, the intentional subject that *must* exist, that necessitates a particular set of possibilities (84).

Lundberg and Gunn's problematization of the agent stems from this anxiety: for Gunn, the anxiety over agency is a mediation of how we think about the possibilities of agential action, that "[i]nsofar as fantasies offer, simultaneously, a frame within which to exercise agency . . . our contemporary anxiety about the rhetorical agent is a generative or productive scholarly neurosis" (Gunn 2004, 19). The mediating effect of fantasy at once gives us legs from which to articulate forms of action. That is, the "gaps" upon which Rand bases her theory of agencies can be seen as gaps of mediation, in which the novelty of agency arises from an understanding that there is no actual connection, but a response to and uptake of rhetorical practices.

Mediation being a key term here, I think, works precisely because we have no direct link to the reality of agency, simply the effects of acts from which we can deduce particular kinds of agency (and not agents). Thus the media through which effects are consumed construct the ways in which we think about agential potentiality. As Friedrich Kittler (2009) writes, "a medium is a medium is a medium. Therefore, it cannot be translated" (265). The "cold reading" of effects do not come from the extraction of content from the observation of these effects. If we follow Lundberg and Gunn, the co-construction of the possibility of rhetorical agency comes through mediated engagements with agencies that are only interpreted through effects. But as the mediation of these effects produces the necessary framing for exercise of any agency, we should also recognize that the movement of "meaning" through mediating agencies is not simply a form of translating those meaningful performances through those agencies (the transcendence of communication). Rather, the movement of one medium to another, of articulating the function of something within the logic of another, requires that we let go

of the universal translatability of meaning as a metric for studying effect, success/failure, or persuasive capacity. Mapping Kittler's distinction between translation ("the exclusion of all particularities in favor of a general equivalent") and transposition (the serial movement of the logic and relations of a medium into another) locates intelligibility not simply within a general realm of signification, but also within a material and logical realm of mediating concepts and/or technologies. Thus communication, seen as moving through the "channel" of language, breaks down when it is a function through and of mediation (266).

(4) That rhetoric as a productive art must call upon a subjective capacity for agency that allows for its teaching, training, and/or transference. That is, rhetoric has as its core a pedagogical project that maintains its legitimacy as a discipline, and it is tied to the assumptions of the first three claims above. This teachability, in fact, suggests its own opposite (the "non-artistic" critical approach to rhetorical studies) and the ways in which this opposite only comes about through a loss of the civil, pedagogical project of rhetoric (and the loss of rhetoric's "birthright").

Rhetoric and the Digital

From the above, I hope to construct a counterpoint to the prevailing assumptions about subjectivity and agency, viewing this opposition through of a contingent politics that favors an indeterminate relationship between agencies, actions, and effects emerging from mediating technologies. But I further ask if there is a platform here in which my counterpoint engages with the possibility of civic rhetoric, or if nothing else a rhetorical pedagogy of media not committed to the humanist project. I illustrate some connecting

work within the realm of digital rhetoric (an area of study in itself concerned with rhetoric and its uptake through mediating digital technologies). And while digital rhetoric as a field is immense and only growing larger, a few examples can illustrate a movement I want to follow. James Zappen (2005), in his essay "Digital Rhetoric: Towards an Integrated Theory," articulates a few approaches to digital rhetoric that have emerged in the field. Zappen breaks downs these approaches within a few categories that show how rhetoric works in or through digital technology. First, he points towards the ways in which "traditional rhetorical strategies" are reconfigured or function in digital spaces (319). The authors that Zappen cites here draw from traditional Greek rhetoric, and situate rhetoric as a method of communication rather than just persuasion. Zappen also discusses how technology constrains and enables the enactment and transformation of rhetoric in digital spaces, and how identities and communities are formed symbolically in these spaces (321-322). Zappen's takeaway here is that technology mediates rhetoric, but that rhetoric itself is something that exists outside of technology to some extent, something that travels through or is transformed by technology. While Zappen is more than open to how rhetoric is transformed through technology, the extrapolation of rhetoric as a theoretical reality outside of technology draws Zappen to think of rhetoric in terms of discrete discursive objects that travel through technology rather than something emergent from it. This theorization of rhetoric leads to the fashioning of rhetorical theory (rather new theory or "transformed" theory) in terms of "spaces," "communities," and "identities." While still commensurate with the idea of rhetoric as political, as expressive, his move towards an integrated theory fashions existing rhetorical notions of subjects and

agency into new digital modes of communication. Thus we get terms such as "spaces," "communities," and "identities" as defining paradigms for communication online.

Other scholars attempt to move away from some of this language, seeking out new ways to think about how rhetoric is studied in produced through digital media. Collin Brooke (2009), in *Lingua Fracta,* discusses the five classical canons of rhetoric and re-articulates them within the framework of a digital, networked culture. These canon, once seen as the origin of rhetorical education and preparation, are an attempt to think about the generation and circulation of rhetoric through digital media (211). Brooke's method is similar to Byron Hawk's (2004), who in "Toward a Rhetoric of Network (Media) Culture: Notes on Polarities and Potentialities" shifts traditional rhetorical concepts such as Aristotelian ethos, pathos, and logos into the language of digital communication technology in order to discuss the new ways in which rhetoric functions in network culture. James Porter (2009) situates rhetoric and delivery (another canon) within digital technology through a discussion of embodiment, circulation, and access.

This configuration of rhetorical theory within the digital acknowledges, to some extent, that "transformations" of rhetoric but does not entertain the question of how the underpinning assumptions of rhetoric might be fundamentally different. In looking through "spaces" and "identities," there is a work of seeing rhetorical assumptions of audience and rhetors as discrete categories that situate rhetoric as a communicative act through traditionally-know discourse. As an interesting move away from this, however, we might look as Elizabeth Losh (2010). and her definition of "digital" rhetoric. In her

book *Virtualpolitik*, Losh discusses four definitions of digital rhetoric that she relates through their definitions themselves:

1. The conventions of new digital genres in everyday discourse,
2. Public, political discourse disseminated through digital technology,
3. An academic discipline that studies rhetoric in digital technology, and
4. The discipline of information science and its discussion of information integrity,

   decision-making, and uncertainty (47-48).

In working through these definitions, Losh responds to Zappen's final argument that digital rhetoric is a theory built around a particular set of discrete rhetorical objects, by saying that a general theory of digital rhetoric could be developed, if rhetoricians are willing to recognize the movement through these various definitions. In particular, the definition (4), which draws from information science and cybernetics to discuss the Shannon-Weaver model of communication (which works through questions of signals, noise, and information integrity), offers ways in which to refine research questions that stem through the first three definitions. That is, para-disciplinary conversations that move through the sciences, the humanities, and many other areas of knowledge will determine how we think of rhetoric in digital technology, not simply how we think of rhetoric itself, alone, within its own disciplinary confines (98). And while Losh is still articulating a digital rhetoric that thinks back to classical determinations of communication and persuasion, her last definition opens up an interesting avenue for inquiry. I look at this as an invitation to think about the media of communication in rhetorical studies not as the

carrier or transmitter of communications, but to also think about how technology function rhetorically (that is, to look at how rhetoric is manifest from questions tied to the technicity of its deployment).

Losh's definition, I suggest, leads us to a question of medium. Media theorist Jussi Parikka writes that anything can itself be a medium: a medium of transmission is not *a priori* decided as inert technology that stands in between two intentional communicators. Media is often coded, biological, effectual methods of co-constructing worlds through communicative means. Subjectivities and agencies is this not entirely contingent (they emerge through a play of media protocols) , and a rethinking of the production of agencies, begs that we understand agency as not necessarily hegemonic or resistive, or necessarily arising from a point of articulation, but as a product of media logics. In "Protocol, Control, and Networks," Galloway and Eugene Thacker (2004) argue that the strength of protocol and networks is that "the problem-solving process is not dependent on any one problem-solving 'agent' but that the solution . . . arises from the context of distributed regulation" (18). More apropos to rhetoric, a protocol is written as a text, certainly, and there is an audience who will read such technical documentation and enact it, respond to it, and so one. But the product of a protocol, the formations of mediated communications that continually iterate over and through one another, do not fit within the rhetorical construct of text, audience, author. It would be a mistake to further call the users of the Internet an "audience" even as they are continually moved or articulated through the effects of the various protocols that comprise its formation.

Gender, Queer Computation and Collaborative Labor

The links I seek to create here are those between the user and the creator of software as co-constituents in a cultural practice that also involves an active and culturally significant technology. If the anxiety over rhetorical agency is embedded within a loss of potential political or pedagogical importance, then a posthuman account of technology allows for such purchase. However, it is also quite easy to allow posthuman rhetorical studies of media to alternatively bracket cultural categories of gender, race, or class behind a more broad and abstract notion of network or computational culture. Technology becomes a master discourse. Once it is broken from a technological determinacy and grounded in a space of open theorization, it does not necessarily lose the luster of its novelty or uniqueness. But technology as a horizon of new social interaction often belies the labor-intensive realities of the creation of technology, and often does so by focusing heavily on either technology's capacity for representation (either in media or in its creation) or in its capacity for abstracting its cultural roots.

Following this, I want to introduce a rhetorical approach to networks and technology that attends to the gendered underpinnings of the history of digital media. Chapter 2, then, initially constructs a foundation for a rhetoric of networks as rhetorically meaningful structures through a theory of queer protocol. Through an intersection of queer and gender theory (specifically focusing on Judith Butler and performativity and Halberstam's notion of queer failure) and critical code studies, I will discuss Zach Blas's "transCoder" programming SDK and Julie Levin Russo's *Slash Goggles* to illustrate the connections between computer code and the performance of subversive "disidentifications" through digital technology. Jussi Parikka (2010) writes that media are

constituted by "forces imperceptible to human understanding but glimpsed through their effects only—the logic of algorithms, calculations, and voltages unreachable in itself, yet continuously mediated and affecting the bodies of humans," and as such the work of forming subjectivities and agencies must work through multiple registers of communication outside those prescribed by humanist models of discourse (199). The primary question here is how practices intrinsic to computer programming—specifically the technique of "abstraction"—simultaneously represent an efficient and effective approach to software development, and also a historically gendered form of digital labor. I discuss abstraction as a performative discourse in that it creates the interface mechanism between previous and immense histories of labor and the potential for new technological development. The significance of this is that the historical relationship of technology and gender is not simply constructed in a culture external to the technology, but is embedded in the technology itself. Blas and Russo not only demonstrate potential disidentifications to code as a totalizing discourse, but offer a way for scholars to discuss methods for engaging both the cultural and technological aspects of that discourse.

Chapter 3 works through a study the BitTorrent protocol. and how this functions as a way to think about agencies outside their grounding in determined subjectivities. BitTorrent offers a new way to share files that refashions older client/server models of peer sharing by exploding the model across a paradigm of swarms. The "swarm" is therefore an abstraction of underlying Internet protocols like TCP/IP that builds from the technology to accomplish a task more efficiently. Devised in response to the closure of various file sharing services, and the prosecution of thousands of individuals within these networks, I illustrate the overlapping concerns of copyright interests and how Bittorrent

embodies a refiguring of these concerns. This protocological examination discusses the formation, production, and mutation of agencies and subjectivities through the realities of digital information (copying, transference, and distributed storage and file maintenance) and the new legal and distributive formation Bittorrent constructs. Likewise, the concept of the swarm is not without its own political implications, and I discuss the notion of a swarm within a larger context of collaborative labor to illustrate how BitTorrent and swarm technologies operationalize existing logics of control and hierarchy. Here, I link these logics to collaboration and gender dynamics discussed by rhetoric scholars such as Karen Lunsford and Lisa Ede, as well as "global hegemonic masculinities" theorized by R.W. Connell (1998). Then, through José Esteban Muñoz (1994) and his theory of disidentification, I introduce a possible complimentary mode of protocologial work suited for rhetorical network analysis. Munoz writes that

> Disidentification is about recycling and rethinking encoded meaning. The process of disidentification scrambles and reconstructs the encoded message of a cultural text in a fashion that both exposes the encoded message's universalizing and exclusionary machinations and recircuits its workings to account for, include, and empower minority identities and identifications. (31)

I argue that the digital disidentifacatory move of the Bittorrent protocol is precisely in the way in which it exposes and refashions the encoded meanings of the other protocols it contends with in order to "crack the code of the majority," albeit in a different context and scale. Building from Chapter One, I argue that digital disidentification is a performance that allows us to think about agencies produced by digital media outside the

horizon of the humanist, deliberative subject without necessarily giving up a framework for political engagement.

Chapter 4 explores a third case study, this time in the form of cryptocurrency, namely Bitcoin. Bitcoin takes the swarm mentality of BitTorrent one step further my mobilizing its logic into the operational value of trust. Trust, being centered in a protocol (rather than financial institutions) places Bitcoin in the ironic position of critiquing old and hegemonic technology while reiterating many gendered norms of finance and abuse across Having moved from identity formation to the question of efficacy and action, I explore how these questions address a seemingly non-discursive regime. The search for meaning, identity, and action within these networks is queered so that Bitcoin, as a disidentifacatory performance (it uses many of the techniques associated with Bittorrent), muddies how audiences and rhetors function as theoretical concepts, focusing instead on the iteration of the Bitcoin network as rhetorical performance itself. Thus “iteration,” in the Derridean sense, is played out not only through the network’s own construction of itself and its point of articulation, but through the ways in which new forms of persuasive action and effects are produced. Iteration and disidentification work with Galloway's protocol analysis to form a rhetorical analysis of protocol and emergent agencies, including the possibilities for "queering" these networks and how we think about the circulation and production of rhetoric.

Chapters 5 explores the theoretical implications of my case studies, focusing on the movement in technology through rhetorical concerns of subjectivity, agency, and political engagement. I argue, through the analysis of these three case studies, the ways in which subjectivity, agency, and thus politics are products of various technological

mediations and techniques that include, but do not require, normative politics of agency or rhetoric. Instead, I show a particular mode of networked rhetorics not looking towards how networks *circulate* existing rhetorics and agencies through classical civil and civic mechanisms, but rather looking towards a rhetoric *of networks and circulation* that accounts for an ethical, accountable subjectivity or agency based on contingent criteria of action mediated through digital technology but still capable of engagement and change. I suggest that a rhetoric of circulation understands the generation of social and cultural categories within digital communication technology, and further argue for rhetoric's place in the study and theorization of such technology.

The result is an approach to communication technology that focuses on the aspects of that technology that tend to evade scrutiny in terms of social or cultural critique. Media theorists have suggested that the actual workings of digital technologies are lost on us so long as we maintain a study of surface or interface phenomena. I would also suggest that it is difficult to necessarily define where interfaces end and actual technology or control mechanisms begin. However, I argue that in the case of source code and computer technology, it is not so much the task of media and rhetoric scholars to find the origin of technocultural phenomena, but to better understand the continued recreation of that phenomena in different communicative contexts. The performativity of gender and technology as a method of collaborative labor in circulation is one such context.

Chapter 2
Computational Performativity and Queer Code Practices

Introduction

Digital media is not gender neutral. While many arenas of software development posit a gender-blind meritocracy of progress and achievement, the historical and epistemological assumptions of the field contradict such utopian visions. As Wendy Chun (2011) writes, the original computers were in fact women, literally computing problems given to them by (typically male) mathematicians and scientists, plugging away on 1940s state-of-the-art military-grade hardware (31). This man-woman-machine relationship is the precursor to the development of software, and as men stepped into the burgeoning field of "programming," so to was digital media bracketed as prior to or outside gender, effectively moving it from the clerical realm of hardware manipulation into the creative and intellectual realm of computer science.[1] Since women, in many ways, were software before there was software, scholars can see digital media as an inherently gendered entity develop methods to study technology as a productive and performative aspect of cultural epistemologies. Not only does code represent the instructions that eventually come together to form software, but it also expresses a set of knowledge-producing practices that join programmers and users through a plethora of technological enterprises.

[1] The six original ENIAC programmers were Kathleen McNulty, Betty Snyder, Betty Jennings, Marilyn Wescoff, Frances Bilas, and Ruth Lichterman. See W. Barkley Fritz, "The Women of ENIAC," *IEEE Annals of the History of Computing* 18, no. 3 (1996), 15.

In this chapter, I argue that one such epistemology, that of “mastery,” exemplifies the performative and gendered nature of software and code practices. I situate programming as a profession of software production, the result of performative norms of humanistic mastery of discourse, technology. Towards this, I extrapolate and refine the concept of "computational performativity," drawing from media theorists Katherine Hayles and Wendy Chun to put forth that computational performativity *is* gender performativity. Rather than see code as a representation of meaning through representations of texts or processes, I utilize several code artifacts to show that software is more accurately examined as an ecology of development practices that structure how certain norms circulate through software. These practices can be traced through the distinction between code as a text (source code, algorithms) and code as software—as made from the relationships of multiple, often hidden processes. Inspired by Judith Butler’s (1997) assertion that "[t]he linguistic domain over which the subject has no control becomes the condition of possibility for whatever domain of control is exercised by the speaking subject," I trace how computational performativity functions as a technological production of structural masculinity through software (28). Moving away from representation as the seat of meaning in code, I see the reading of code as a formal tool to discuss how gender works through what I call “contexts of complexity.” Complexity, in software engineering, is a constant problem and mechanism that requires that programmers strategically hide complexity to produce further, complex software. To read code as hidden and hiding, always embedded in contexts beyond its immediate application, is essential to understanding how assumptions of discourse propagate through that hiding and production.

To illustrate my vision of computational performativity, I examine two related digital artifacts: Julie Levin Russo's gender-challenging *Battlestar Galactica* fan fiction *Slash Goggles* and Zach Blas's digital queer art project *transCoder*. A fictional "pseudocode", *Slash Goggles* is an algorithm meant to bring together elements of computation, fiction, and gender critique. *Slash Goggles* mobilizes computer programming structures and technical communication to present a theoretical piece of software that operationalizes sexual identification through characters in the show. The programming syntax used as the "language" of *Slash Goggles* is obtained from *transCoder,* a digital art project that synthesizes software development techniques and queer gender theory from thinkers such as Butler, Luce Irigaray, Michel Foucault, and Donna Harraway. Using *transCoder* as a template for *Slash Goggles*, Russo develops her project within a set of computational norms and fictional contexts of complexity that perform traditional software engineering practices. As a consequence, *Slash Goggles* implicates her critique and her fan-fiction within a larger critique of digital circulation and sexual identity.

The move from a template library of code in *transCoder* to a realized procedural critique in *Slash Goggles* demonstrates the performative move of software in iterating social and computational norms, and the potential for gender norms to iterate as well. In making this argument, I proceed in three parts. First, I define the concept of computational performativity through *transCoder* and *Slash Goggles*. I show that the former structures the latter through what Hayles (2005) describes as the work of "revealing and concealing" in software, or the ways in which the "'brute' lower levels" of

computation are concealed to engender higher-level order through a mechanism of "abstraction" (54) Second, I describe how the relationship between abstraction and complexity in code is a performative aspect of the circulation of a masculinized epistemology of mastery, in which the subject position articulated by programming as a practice rests on assumptions of expertise, individuality, and command. Third, I demonstrate how their queer critiques of code open the way for a rhetorical approach of computational performativity by way of the discursive potential of what J. Jack Halberstam defines as "queer" failure in code. Abstraction as performativity and queer failure allow us to recognize code as masculine epistemology, to critique that epistemology, and also recognize the productive aspects of that epistemology.

Computational Performativity and Contexts of Complexity

Describing code as "performative" presents a comparison of code and (or as) language, opens up the question of code as discourse, which for some is a contentious analogy to make. Because code is not "natural" (ie, human) language, some overwhelmingly see that it is immediately constrained by its determined, machinic limitations as compared to natural language. Hayles (2005) argues that we cannot describe code in the same way as a natural language because code has a more rigorously defined referent in the form of hardware and execution structures, of esoteric commands, binary digits, code libraries, and electrical pulses--referents that have no antecedents in natural language (54). For Hayles, the "effects" produced by computer code invoke, through the execution of code as part of software, an unintelligible interlocutor in the form of the machine. It is not uncommon for critical and rhetorical scholars to situate code as a symbol system to

interpret, and in some respects this is not an entirely unproductive avenue of inquiry, even if it only provides a partial image of what software actually is. Robert Cummings argues that the purposing of code as a text allows us to better understand how we write for human audiences, that there is a communicative relationship between machine structures and human writing. The machinic elements of this equation (the programming language, the software) are--even though they structure the entire activity-- peripheral to the actual activity of writing the code (which, ideally, adheres to relatively understandable paradigms of rhetoric, composition, and pedagogy). Likewise, code studies scholar Mark Marino (2002) suggests that we can “read” code (or pseudocode) like a poem, "a text, a sign system with its own rhetoric, as verbal communication that possesses significance in excess of its functional" utility (Marino). This can be a strict representational look at the code (what does it say), or, as Ian Bogost (2007) or Kevin Brock (2012) suggest, we can look at the rhetoric of an algorithm through the procedure outlined by the code or expressed by the software (what does it do, or what does it ask the user to do). Algorithmic analysis coincides with the notion of procedure or process as discourse, an idea explored through digital realms such as games and software. The queer implications of games and software have already been explored to some extent. Edmond Chang (2015) argues that games can offer both subversive and normative constructs of gender in gameplay through choices offered to the player (9). Bonnie Ruberg (2015) further argues that such processes argue opportunities to evoke affective responses outside emotions traditionally ascribed to game play--emotions such as sadness, fear, and disappointment (111). Such examples acknowledge something other than language or representation as meaningful aspects of digital media, and such aspects are also present in

code. Since coding as a practice does not necessarily map on to human-centric practice of writing, the limits of a textual interpretation of code are revealed through the execution of that code. But, if performativity is less about effects in audiences and more about what Butler (1993) argues is not "a singular or deliberative 'act,' but, rather, as the reiterative and citational practices by which discourse produces the effect that it names," then we can implicate the network of relationships that make code work as software (xii).

Software is not "an algorithm" or code, but a network of code libraries producing contexts of complexity for execution that cannot be limited by an "audience-interlocutor" model, nor reduced to a mediation within such a model. In the examples of *Slash Goggles* and *transCoder*, computational performativity is demonstrated through the artwork of queer digital activists' representations of code and software, and opens up to an investigation of how computational performativity can produce gendered effects. Because Hayles qualifies the notion that code is a language by implicating the computer itself as an audience, she qualifies code as a mediation of intention and logic between humans and machines. She accordingly resists a Derridean notion of language's infinite citationality and iterability--since code is tied to a knowable context (that of execution, of the machine), at some point "signifiers must point to signifieds," and commands must work within a tight constraint of "correctness" to execute (Hayles 48). Derrida's description of language as always already a citation of previous utterances-- utterances that iterate outside of the presences of their author--ends logically at the artifice of hardware. Therefore, the *writing* of code is not directly connected to the execution of that code through that act of writing: there are intermediary steps during the computational process

that programmers and writers do not engage, which seemingly ends the endless play of signification present in natural language.

Within this discussion of code and execution, note that when Cummings, Marino, and Hayles refers to "code," they more specifically mean "source" code. Source code is the code that programmers write, modify, and compile as part of the software development process. This differs from machine code, which is expressed in binary or hexadecimal numerical form and corresponds directly with the processes occurring in computer processing and memory. Non-technical audiences are familiar with source code through representations of programming in popular media, with focused hackers, spies, or scientists typing away while lines of mathematical-looking statements scroll down a screen. While esoteric and difficult to understand, it is relatively translatable because source code more closely resembles a natural language. The simplest example of executable source code for most programming languages is the "Hello World" program, often used as a beginning lesson in programming textbooks and tutorials. "Hello World" in the C programming language looks like the following:

```
#include <stdio.h>
int main( ){

        printf("hello, world\n");
        return 0;

}
```

While the meaning of the commands evades non-programmers, it takes very little effort to translate: within a main function (int main) it prints (printf) the sentence "Hello

World” plus a line break (“\n”) to the screen. This is, for all intents and purposes, a complete program. Similarly, *Slash Goggles* is a representation of source code:

Figure 2.1 Slash Goggles

This code, while dense and probably confusing to most, still provides a foundation for “reading” some of the intended meaning of the author. Unlike “Hello World,” *Slash Goggles* is “pseudocode,” or non-executable text written as code in order to map the logic of an algorithm. Pseudocode represents a sort of conceptual, non-executable “draft” version of a future piece of code.

Russo (2008) posted *Slash Goggles* to her Livejournal blog *The Archive2*, described by Russo as "experimental fan works for *Battlestar Galactica* season 4 and beyond," The blog organizes its contents around a shared interest in *Galactica* and the identity-challenging themes present in the show. *Slash Goggles* addresses *Galactica's*

themes directly through the conceit of the Cylon--human-like cyborgs who experience emotions like pain, fear, and sexual desire, and who inevitably model their identities upon (and serve as model for) their human counterparts.

The blog post ("the *Slash Goggles* algorithm") contains the *Slash Goggles* code, a description of that code, and a series of images from the series, photoshopped with thought bubbles that reflect what the outcome of the algorithm is. As such, Russo's fan fiction more closely resembles a documentation page for software than a fictional narrative, including descriptions of how the code works and an implementation guide. This documentation describes *Slash Goggles* as a program that "establishes, based on visual data, the variant sexual preferences of human subjects," a program that helps Cylons understand human sexual orientation and and "improve HCI" (Human-Computer Interaction) (Russo).

The fictional language *Slash Goggles* portrays ("Cylon COre Language--CyCL) functionally and textually represents scripting languages like PHP or JavaScript. It starts with (1) a function declaration ("function slash_goggles($desire)") that signals the beginning of the algorithm by way of naming it. Functions are, much like basic algebra, names that "stand in" for larger computational formulas that involve variables. In this case, "slash_goggles" names the function, and defines what variable data it takes as input ($desire). Words beginning with a dollar sign ('$') designate variables. Like math problems, variables take a value such as a number or a string of words assigned by the program in order to do work with or on that value. Following that, a series of sub-procedures accomplish component operations: checking activation status (2), defining subjects (3), verifying data (4), parsing that data (5), and using that data to computer

some sort of gender identification (6). *Slash Goggles* takes as input the variable "$desire," which immediately implicates some sort of value that quantifies or represents input that has been assigned to represent "desire" by some other program. Then, in a series of smaller sub-procedures, the algorithm accomplishes a set of tasks seemingly integral to the identification of sexuality. The "exactness" of code in expressing a procedure through concrete statements and quantifiable results demands a specificity that *Slash Goggles* both leans upon for its rhetorical effectiveness and, at the same time distorts, through linguistic ambiguity.

Slash Goggles critiques and destabilizes both gender and computation as purely mechanical or determined categories, and questions the distinction between the two by staging sexual identification as a procedure. Since the code represents a procedure, Russo also includes some fictional "output" of the code. Images from the show featuring human characters, seemingly viewed through the point of view of a Cylon, are given thought bubbles that articulate that character's sexual desires, frustrations, and complexities. Figures 2 and 3 represent the results of the algorithm as the Cylon characters (in first-person point of view) gaze upon the human characters.

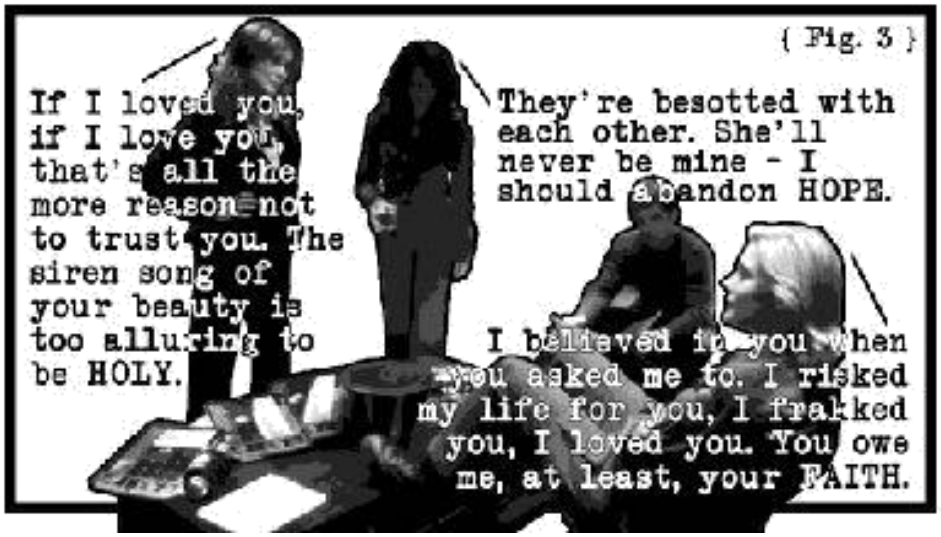


Figure 2.3 Cylons Using Slash Goggles

Figure 2.2 Cylons using Slash Goggles (2)

Within the framework of her fan-fiction, Russo suggests that this algorithm is similar to human recognition of sexuality ("gaydar"), and explains that *Slash Goggles* represents the procedural work of ascertaining a shifting, fluid sexuality that is drawn from, but not determined by, social dynamics and negotiated subjectivities.

The *Slash Goggles* operation/algorithm describes the negotiation of sexual identification. Here, we can start to look for textual clues to interpret what *Slash Goggles* attempts to express as a critique, because the ambiguity between the language of the code and its use as code helps constructs the potential meanings of *Slash Goggles*. Readers unfamiliar with code may still pick up on references to sexuality, gender, and queer epistemologies in phrases like "theCloset," "buggery()" and references to the body, identity, and gender. Those familiar with coding will also notice the programming structures that meaningfully situate those references. For example, in "defining subjects" (Section 3 in Figure 1), *Slash Goggles* iterates over each element in $humanform (which, as suggested by the code, contains at least one, if not more, discrete elements) and assigns a value based on visual data ascribed to a variable "$body."

The rest of the algorithm follows suit, and calls to functions that reference "theCloset," destabilization, and performance suggest a similar reliance on contextual, yet open and circulating, interpretations of subjects and their sexualities. As Eve Sedgwick (1990) argues, the "closet" is "a performance initiated as such by the speech act of silence--not a particular silence, but a silence that accrues particularity by fits and starts, in relation to the discourse that surrounds and differentially constitutes it" (32). The use of "theCloset" as a code term suggests a self-orientation (is the Cylon subject 'in' the

closet? Can the Cylon perform or express desire? How does this orient the Cylon towards the subject of their view?) and a cultural orientation constructed by the ability to speak or the necessity of remaining silent and/or hidden. More directly, "theCloset" plays a part in construction of sexual knowledge for the Cylon--if “theCloset” is 'null', meaning without value, then the algorithm begins openly constructing data based on sexual traits gained from interpretation of the "$body". The computation of a sexual identification of the other is already based on the viewer's interpretation of the other, which is inherently incomplete and limited, but informed by the available (and, conversely, the unavailable) information.

The algorithm queers sexual identity and identification as well as computation, both in its language and its formal structure. The wording of commands as recognizable ideas in gender and queer theory are mobilized within programmatic constructs like conditional statements, iterating loops, and variables to express a structural critique of sexual identification. By taking a seemingly determined media format (computer code) and using it to demonstrate an ambiguous process, Russo suggests that it simply is not enough to say "sexual identification is ambiguous" or that "gender is dynamic and interpreted," but instead calls for the reader to walk through the difficulty of operationalizing such a task. As Erin Davis writes, "[i]dentification occurs within a social regime of normative expectations and guidelines that shape everyone's possibilities for self-representation," and that gender identity is not “static, but it is also not unbounded.” Literally, in this case, identification is bound by structure and syntax. In this way, potential identification is implicated in logical structures and interconnected activities not entirely expressible through procedural understanding. Values, epistemologies,

foundations of activity and knowledge, are all continually (re)expressed in these code relationships, even as they are hidden. The expression and critique of sexual identification is linked, through the use of the algorithm and code as medium for such expression, to a resistance to any notion of an algorithmic reduction of sexual identification as a process and to the reduction of a process as something rote or determined. The underlying performativity connects these approaches as critiques of code and sexual identification, with shared concepts of determination and agency. The connection between these two performative moves is linked through the underlying performative nature of the computational system, and in particular through the gendered nature of that system.

Much as Russo situates identity and identification as a procedure, the underlying work of code implies a realm of computation and execution in excess of a code's meaning. We must therefore situate code as a performative structure outside of strict interpretations of author-intended meaning. Taking source code as the origin or center of meaning, according to Wendy Chun (2008), erases the distinction between code and execution, conflating the meaning of that code with its effects in computer execution (303). This represents how, according to Chun, how source code is "fetishized" by scholars as the seat of meaning (Chun 309). To fetishize is to erroneously assume that the full meaning of the code is exposed as the source is exposed. Performative code, however, places something like "Hello World" or *Slash Goggles* outside its author as the seat of meaning. As a given piece of software is most commonly tens of thousands (if not millions) of lines of code, the functioning of that software is predicated not simply on a given algorithm, but on the relationships between algorithms in different code libraries

and how user input circulates between their execution. If execution and code are separate and under erasure, as Chun argues, meaning cannot be invested strictly in the source code, but within the erasure of the underlying execution of that code--which includes the hiding of hardware and the hiding of underlying code libraries and algorithms. The technique of hiding execution is known in computer science as "abstraction," a method where complex computational processes are hidden or "abstracted" from users behind simpler interfaces to facilitate easier, more streamlined programming.

Abstraction is the mechanism of computational performativity, because this erasure allows the development of complex software from already-existing complexity. A good example of this is the software development kit (SDK). The idea of an SDK harkens back to the previous discussion of hiding complexity: an SDK provides programmers with code interfaces that allow them to produce software for complex platforms. A good example of this is the development of software for mobile devices like smart phones. Android and Apple iPhones are incredibly complex machines, and the code necessary to do even the simplest task (like draw a window to a screen, or connect to the Internet) would be monstrous if each app on that platform required programmers to do each from scratch for every program. Therefore, these platforms have dedicated SDKs, written in a given programming language (Java for Android, Objective C for iPhone) that hides the complexity of hardware and software interaction so that better, more complex apps can be built faster and more reliably. The code required to have an Android phone draw a window to the monitor requires mathematical computations to determine size and orientation, contents, and the very context from which the window comes into being and presents itself to the user (including where to store this in memory, how to toggle

between user contexts, etc.). When every task requires hundreds, if not thousands, of procedures, suddenly a simple program becomes almost unthinkably complex. However, with the SDK, the command to create a window and start an app is reduced to about a dozen lines of code, none of which require an app developer to manually manage hardware, data resources, or the operating system Thus an SDK is an abstraction of the underlying complexity of the phone that allows for the quick and reliable deployment of software for that phone.

Abstraction points us towards the performative because it incorporates regulatory practices outside of the code that we write, and these regulatory practices often consist of assumptions about how people work and communicate. In "The Performativity of Code," Adrian Mackenzie (2005) argues that software can represent "a form of collective agency in the process of constituting itself" (73). Using the operating system Linux as his example, Mackenzie writes that the ¨ongoing constitution is performative with respect to the efficacy of Linux as a technical object and with respect to the fabrication of Linux as a cultural identity" (73). For Mackenzie, Linux regulates its own constitution through a series of norms of practices that are within its own code, which then circulate through the social body of developers that utilize it. This echoes Butler's (1990) argument that performativity is a production of subjectivity and sex *through* the iterative power of discourse (133). That is, it

is not an act or performance accomplished by some actor, but the continued (re)production of norms, rules, logics, a reproduction that becomes naturalized or seemingly necessary as it materializes and rematerializes itself. Software "would have to be understood not just in terms of the meanings ascribed to it, or in terms of its effects on

the movements of data and information in communication networks. Rather, it would be an objectification of a linguistic praxis" (Mackenzie 76). Thus software must contain within itself the logic of its own reproduction. However, to stay close to Butler rather than, say, Derrida, I turn to an example of abstraction that retains Butler's investment in gender: Blas's *transCoder*.

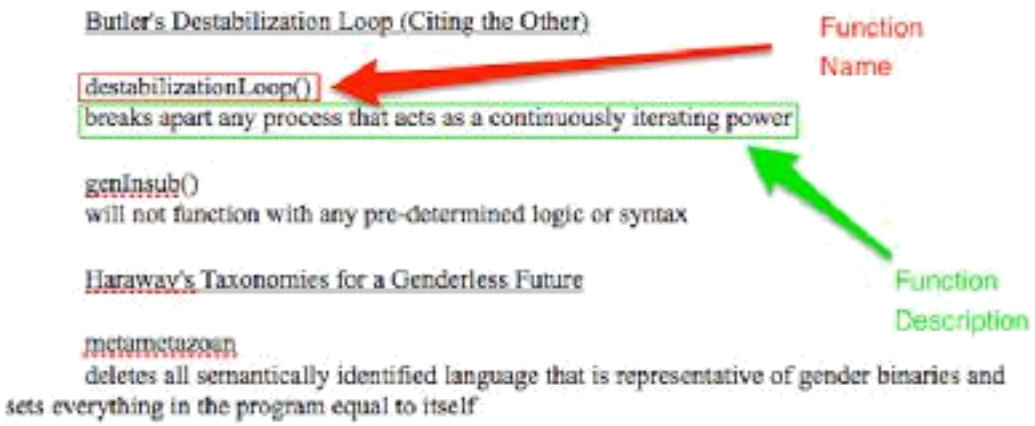


Figure 2.4 transCoder

Burned and distributed on CD-ROMS or as packaged files available for download through Zach Blas's Queer Technologies website, the *transCoder* package appears a simple collection of unordered .txt files with names like "about.txt," "libraries.txt," "compiler.txt." *transCoder,* like *Slash Goggles,* is pseudo code, but whereas *Slash Goggles* mimics an algorithm, *transCoder* more closely resembles an SDK. Marino (2012) describes *transCoder* as a "provocative kit" that "uploads counter-cultural ontologies (or anti-ontologies) into the normalized logic of software. He is transcoding theory into a programming language" (189). The accuracy and importance of this description is evident in that, unlike the source code of *Slash Goggles,* what we see here is not an algorithm. In fact, without some prior knowledge of how code functions, we might not have any clue what we are looking at, outside of some recognizable names and

references. In the above example, Blas structures a Butlerian concept of iteration and power through the programmatic statement "destabilizationLoop()," and a play on speech act theory with a series of "executable speech acts" (such as “iDo()” and “exe()”). *transCoder,* as an SDK, does not provide a series of code statements that represent the procedures underlying the command “destabilizationLoop().” Instead, the user is given a short description on how “destabilizationLoop()” *should* be used. Or, it tells the user what it does, and not how it does it. "destabilizationLoop()" "breaks apart any process that acts as a continuously iterating power", which implies Butler's assertion that gender and sex are products of regulatory regimes but, through changes during the iteration of such regulative norms (or loops), resistance to such regimes can emerge (Butler 1990, 130). The ambiguity of meaning, however, emerges from *transCoder* because meaning is only "implied," as there is no actual code, only the promise of code functionality, because the underlying complexity has already been abstracted from the user. *transCoder* doesn't define an operation, but the potential foundation for the building of operations, a syntax for further operational complexity, out of the interface of promised functionality. So rather than see the work of code as the process of negotiating meaning between a mind and a machine, between language and hardware, we instead see a performativity of code through logical structure.

Calling back to *Slash Goggles,* Russo utilizes "destabilizationLoop()" in a programmatic construct meant to control the flow of a program through decision-making:

```
If (destabilizationLoop('image')){
        $desire = array(mutMutate('identity', 'gender'));
        }
```

Since "destabilizationLoop()" breaks apart any process as an iterating power, the use of an “image” suggests the undermining of stable gender categories constructed through visual interpretation. This code snippet checks uses the "if " statement to check the results of “destabilizationLoop()”. If it returns a “true,” which seems to occur if the loop actually disrupts the iterating power of the image, then the variable "$desire" gains a series of values resultant from the "mutMutate('identity', 'gender');" statement. "mutMutate()" can "connect any number of items to generate hybrid functions, operators, variables, etc.," and because of this, once the iterating power of “image” is broken, then “$desire” contains the possible hybrid connections of identity and gender. All of the operations described above are rendered meaningful through the framework of *transCoder*, in that the use of *transCoder*’s code calls upon a pre-existing coding context. The fact that we know nothing about how “destabilizationLoop()” actually breaks an iterating power demonstrates how my reading can extrapolate meaning from *Slash Goggles,* and how something as seemingly ambiguous as “destabilizationLoop()” still performs within the context of computer code. There is no necessary revelation of *transCoder*’s code, but only the performative structure that *transCoder* creates.

Also note here a few additional lines from *transCoder*'s libraries.txt. file:

> schizoA()
>
> Replicates exponentially and erratically
>
> buggery()
>
> acts upon a function or data set and generates an array of monstrous non-logic mutations. (Blas 2012)

Both of these functions are found in *Slash Goggles*. Russo, using the "schizoA()" and "buggery()" functions, fills in some gaps as to how each functions works (variables, input, etc.) by deploying them in a particular way (allocating value, passing variables as input), but their place within a program is articulated through *transCoder*. When *Slash Goggles* calls "'metatext => buggery('queer,' vBody());" as a command, we may now infer from the documentation that the quality "metatext" takes as its value the result of the "buggery()" function when given the inputs "queer" and the result of another function, "vBody()" (which "detonates a time bomb of radical impurity") (Blas 2012). The result is that "metatext" will contain some value gained through the process of generating non-logics of identitarian non-purity related to queer sexuality, which will then become part of a data structure used to understand sexuality--in this case, the object "$humanform." The movement of structure, meaning, and subject articulation is a product of the citation and reiteration of norms laid out in the library. However, unlike a natural language, the abstraction of complexity, and not the language itself, that serves as the performative engine.

As Chun writes, scholars may fetishize their analyses of source code as the central artifact of meaning, one with a relatively transparent performative relationship to the underlying machine (Chun 2011, 29). Coding *qua* writing thus becomes a translatable practice between the goals of machine execution and audience persuasion. While this may not be untrue--in that coding is, in this paradigm, writing--it places the impetus of the practice of coding within a realm of human intelligibility and as a directed activity between writing subject and technology, one that adheres to already-existing approaches to writing informed by traditional discursive models. Abstraction problematizes this

while at the same time iterating social norms, including sexuality and gender, because abstraction illustrates code as a logical interface between contexts of complexity. These interfaces are code libraries, like *transCoder*: they simplify the use of complex code so that programmers can build more complex structures. Chun (2004) writes that "abstraction both empowers the programmer and insists on his/her ignorance . . . abstraction gives programmers new creative abilities" (38). The work of a programmer is therefore mediated through a series of logical translations across digital media, and this mediation is more or less hidden from the programmer at the site of the production of code. Abstraction through interfaces between contexts of complexity produce programming subject positions and articulates a power structure iterating a particular norm of software production. *TransCoder* and *Slash Goggles* inject queer sexual identification into the performative mechanism to circulate queer identities and world making practices into code itself. Such an injection produces productive discursive strategies for digital rhetoric and queer critique of code.

The Queer Performativity of Code and Subverting Mastery

What becomes evident here is that the productive mechanism of computational performativity is no more determined by rote and determined technology than gender by rote and determined biology. In regards to gender performativity, Kendell Gerdes (2008) writes that, "the subject of gender is not in charge, but exposed, addressed by the performative power of essentialist understanding of gender rather than the addresser of it . . . the performative power of gender is its ceaseless materialization of gender," and I would argue that in computational performativity, the same could be said about the

subject of technology (148). In this sense, not only are there constraints to performativity but rather, "constraint calls to be rethought as the very condition of performativity" (Butler 1993, 59). The subject of computation is addressed by the performative power of code through a continual reference to something else -- some structure, some language, some order, that it manifests through. There is no stable "machinic subject" in place as we perform identities in online social networks, through medical databases, and so on. There is only the constant production of those subjectivities, and their transformation across various performative machines. In this way, it avoids the presumption of a metaphysics of the subject where there is a "stable gender in place and intact prior to the expressions and activities that we understand as gendered expressions and activities" (Butler 2010, 147).

From this, a compelling overlap between computational and gender performativities emerges. Both Chun and Mackenzie draw from Butler to argue towards a performativity of code. Both, however, do so more focused on the performativity rather than the gender. If computational performativity is the citation and iteration of complexity across different programming contexts, then I argue that a of masculine epistemology of mastery is structurally embedded within code. Drawing from Chun's account of the history of the ENIAC Girls, a parallel, gendered rhetoric of mastery with computational performativity shows how subjects are formed in programming and software. Leveraging a discourse of mastery within software development links code to what J. Jack Halberstam calls "queer art of failure," a resistance to the determined and correct (masterful) narratives of computer programming. *Slash Goggles*, as an

articulation of *transCoder's* logic, thus suggests that its proceduralizing of sexual identification is also an articulation and critique of mastery as an epistemology.

According to Chun, the earliest days of contemporary programming involved a large, room-sized computer, the Electronic Numerical Integrator and Computer (ENIAC), a military project developed to help calculate artillery tables and formulas related to the development of thermonuclear weapons during World War 2. Built in the Electrical Engineering department of Pennsylvania State University, the computer did not use software. In fact, there was no such thing *as* software. Instead, ENIAC programming was the direct, physical programming of the wires, switches, and other components of the machine itself. There was no "code" distinct from the bare metal of the technology itself. The "computers" were a group of women known as the "ENIAC girls," female engineers who operated the machine directly. These women understood the hardware of the ENIAC and how to instantiate solutions to problems given to them by (male) mathematicians and scientists in processes that could take days or weeks. The objectification of these women under a normative understanding of knowledge and discipline continued through their orientation as extensions of that machine. The kind of machine work the ENIAC girls performed was seen as, itself, rote and mechanical rather than creative of intellectual, and such work was often seen as mindless, unskilled, or "clerical." However, after the invention of "software" and code as we know them today, there was an epistemological collapse of the work of programming and of problem posing and solving, and the image of the lone, brilliant computer programmer came into existence. Chun writes that "programming became programming and software became software when command shifted from commanding a 'girl' to commanding a machine," and following this the

model constructed around “commanding a girl” became a foundation for machine control as well (29). As it became more abundantly clear that programming itself was a skill worth studying, and as men began to apply scientific epistemologies to describe software and programming as practices, then the programmer became a figure of knowledge, expertise, and mastery. No more a cleric of repetitious activities, the programmer was a priest of a “black art,” as John Backus famously described, in which experts handled the arcane through self-driven brilliance and know-how. Nathan Ensmenger (2010) argues that programming thus became masculine when it moved toward categorization as the individual practice of the intellectual “artisan” or the rigorous “scientist” (130).

It is here, in the mistaken assumption that commands may translate directly into machinic code, that the gendered notion of mastery articulates as abstraction and computational performativity. The erasure of complexity is productive in that, in order to produce the conditions of a mastering discursive subject, it erases the invisible and collaborative labor of encoding, translation, and automation in software. Mastery, in this case, is rendered masculine because of its historical nature and because of its epistemological assumptions--the productivity of individual realization through erasure of shared labor. Mastery is the erasure of the intermediary, the investment of creative and productive authority in the articulated individual of code. Abstraction as a technique *does* ease development for programmers, constructing that discursive, subject of mastery, but does so through a willful erasure of the limits and debts of such mastery. Chun notes that “the handing over of power that has been hidden by programming languages that obscure the machine and highlight programming” situates a “master” as an individual, a master not just of the technology but of the epistemology of that technology. The narrative of

mastery is predicated on the strategic abstraction of information to constitute agency within another context. In this case, is constitutes a position of agency in which a system (the programming language) is mastered.

Blas argues that *transCoder* "disidentifies" with technology, he does so to argue that it introduces queer critical theory into a seemingly non-queer space, modifying the circulation of information and identity. Drawing from José Esteban Muñoz (1999), for *transCoder* to disidentify means that *transCoder*

> scrambles and reconstructs the encoded message of a cultural text in a fashion that both exposes the encoded message's universalizing and exclusionary machinations and re-circuits its workings to account for, include, and empower minority identities and identifications. Thus, disidentification is a step further than cracking open the code of the majority; it proceeds to use this code as raw material for representing a disempowered politics or positionality that has been rendered unthinkable by the dominant culture. (31)

However, both *transCoder* and *Slash Goggles* disidentify specifically because, by subverting narratives of code as rote and determined, they further subvert narratives of mastery as epistemological totalities. By utilizing abstraction (and software engineering more broadly) itself as a tactic, rather than as a bracketed epistemology, both introduce into the mastered (or "master-able") space of software the circular, problematic nature of sexual identification, which further, as a performative result, raises and owns the potential and necessity of the failure of that mastery. For Halberstam (2011), failure locates within hegemonic systems of normativity avenues of identification and subversion (89). In an arena like computer programming, incorrectness or failed code is seen as doomed from the get go--it simply crashes and burns. But if failure is a productive performance through contexts of complexity, rather than in the tight confines

of the execution of an algorithm, then failure as a practice can encompass a wider critique of masculine culture in programming. Accordingly, one of Halberstam's theses of failure is that we must "resist mastery" (Halberstam 11). Resisting mastery does not mean simply writing failed algorithms, but writing structural critiques through performative mechanisms that circulate confusion and drives code logics to illogical ends.

Resisting mastery here means resisting the disciplining urge to reproduce, to correct, to maintain, that compels code workers in corporate and academic contexts. Code or software as illogical seems like a contradiction, but only so when the meaning of a given piece of code is invested in the correctness of an algorithm. Bugs, memory leaks, security breaches are all results of the failure of code that escapes correctness, as they all demonstrate the inherent fallibility of the code that always already exists. They are the artifacts of code that escapes its epistemological telos of mastery, which means recognizing the frayed edges of failure that lurks in code inherently, and where this undermines and complicates mastery. Take the "Hello World" example used earlier. The "printf" statement stands as the only real "command" outside of the program structure and header. Printf literally prints a string of characters to the screen, with that string given to the command as an argument. Since it does this, many programmers embed printf statements into their code for debugging purposes, printing warning messages or data values to trace the workings of the program. The implication for the programmer, and more often the new programmer, is that printf will "just print" to the output screen when that line of code executes. New programmers often find out that this is not the case, as development variables and errors in compilation can stall out printf so that it prints at the wrong time or not at all. That is because printf actually works through an output buffer

system, were the string of characters are placed in memory until a specific symbol is reached ("\n"), which then "flushes" (empties) the buffer and puts the data on screen. StackExchange user Toby Speight found this out during an entry level C programming assignment.[2] His example illustrates how failure is part of coding. His program, which calls the printf command, only "printed" at odd times, or not at all, and certainly not as expected. The culprit, in this case, was in the way his development environment compiled the command and executed it--more specifically, in how it handled the buffering and flushing of the data itself. As individuals began to offer solutions on the website, the actual underpinnings of the simple printf command were actually incredibly complex. One user "kliteyn" suggested that Toby try to reset the output buffer with the following command:

```
setvbuf (stdout, NULL, _IONBF, 0);
```

While another user "Kitchi" suggested switching output buffers more generally, using the "stderr" (error) output buffer. Another solution suggested, and one prominent in other areas that I have researched, is the "fflush" command, which immediately flushes output buffers regardless of what else is happening.

Even simple scripts like Toby's or the "Hello World" program are always already built on an ocean of complex code, libraries through which data and system states are passed, transformed, worked on, and returned. Thinking of code through queer failure

[2] See the entire discussion thread at: http://stackoverflow.com/questions/13035075/printf-not-printing-on-console. StackExchange is a forum through which users may ask questions, usually technical in nature, and have them answered by others. It works through a karma system where successful or useful answers are given votes, while unhelpful answers or repetitive questions are, by the rules of the site, ignored or rerouted.

thus recognizes the avenues by which "failure" is less the breakdown of a program or a piece of software, and more a path of subverting or disidentifying with the disciplinary structure of programming through programming. In the above case, it recognizes the tenuous relationship of "mastery" in the face of such complexity, and the productive constitution of mastery as it breaks down and reforms under a necessary admission of ignorance and need for collaboration. Understanding a command like printf as a command for a machine hides the machinic work that makes printf function and, at the same time, the fluid nature in which libraries, compilation, and execution exceed and problematize the completeness of that understanding. The abstraction of the language, of the underlying complexity, therefore constructs a discursive subject that on the one hand can practice a supposed or assumed mastery of code while at the same time including the necessity of collaboration, of consultation. The parallels between the pseudocode examples of *transCoder, Slash Goggles,* and "Hello World" are such that the ghosts of gendered labor and gendered identification in programming epistemologies ask for different, sometimes complementary and sometimes contradictory, approaches. Primarily, the exactness of mastery situates itself as masculine, practiced in code that always cites the abstract structure of its predecessors and reiterates them.

Thus *transCoder* and *Slash Goggles,* outside of their own critique, also offer a critical rhetorical strategy for grappling with gender, sexuality, and code by directly injecting it into the performative apparatus of software development. If we trace how code abstracts and separates contexts of complexity from each other, how it functions as an epistemological performance, then the possibilities for understanding software as a gendered practice are more apparent to us. This is why Ensmenger's claim that coding

became a boy's club is so relevant. For Ensmenger, the building of masculine epistemologies around the practice of code marginalized women out of the field: it required more work and more overhead in terms of learning style. The representation of women in programming has always been a problem, in particular in places where gender-neutral narratives of achievement, merit, and expertise are stronger (many open source development projects fall under this category).

Butler writes that it is this constitutive failure of the performative, this slippage between "discursive command and its appropriated effect" that "provides the linguistic occasion and index for a consequential disobedience" (Butler 1997, 24). Or, as Sara Ahmed (2006) argues, the demands and prohibitions "of fields or grounds for action are generative: it is not that bodies, objects, inhabit structure, but that bodies are expressed, inflected, and oriented by their grounding" (558). Blas's move to define an SDK that abstracts complexity pulls us away from trying to figure out the "code" that makes it all work. This same abstraction allows the movement of meaning by structuring the ability of programmers to build further complexity through interfaces like an SDK. The performativity of gender and the performativity of digital code share an important aspect: that they constitute logics of circulation, one of gender expression, one of software and information. These logics of circulation do not carry meaning but produce it, generate it and its conditions. In networked space, this also includes how bodies are translated across grounds of action, between and through them. These activities, epistemologies, and practices are seemingly played out in a space that should be more strictly bounded. Computer technology, however, is not, and at some point the erasure of distinction between execution and code grounds itself in the mastery of the totality of computation.

But as *Slash Goggles* and *transCoder* show us, there is always already a break in the narrative of mastery: the computer evades us.

Conclusion

Practices of coding, are not represented strictly in terms of words, statements, or commands, but in a structural relationship between these commands and a space of intelligibility that the programmer works in, which is turn constructed through a series of translations and transpositions across differing computational contexts. It is perhaps unsurprising, then, that the question of how gender functions as a discourse within technology is further complicated with questions about how gendered practices materialize in computer code through different contexts. If gendered practices exist at the level of code structure, and not just in the text of a program itself, then code as a discourse relies on structural relationships that are unique to code, that only partially resemble language. If we choose to bracket certain technological questions such as the structure of code as a practice of building software, it may lead us to miss important realizations about that technology. In an interview with *Rhizome*, Blas stated that the work of the Queer Technologies project (of which *transCoder* is a part) is to explore a "viral" aesthetics. For Blas, "[b]y making and mass-producing 'products,' Queer Technologies is able to exist in a variety of contexts without necessarily being identified as an art object . . . They are all designed to to be collective engagements, to be collectively experimented with" (Gaboury 2010). Furthermore, such experiments can help us examine how "heterosexual mathematicians and scientists create models and technologies that are infused with heterosexuality," and how "homosexual desires can

inform and help to materially construct the technicity of objects" Blas and Cardenas 2013, 561).

Blas demonstrates the necessary trans-contextuality of *transCoder* and the fluid nature of its subject matter not only to express gender critique, but to structure platforms for a dynamic critique of computation and sexuality. Even though sexuality and computation seem like disparate topics, Blas looks under the socio-cultural manifestations of computer programming, computer culture, and gender to comment on the epistemological practices of computer programming, which are themselves structured by modes of being that include gendered and sexed practices of communicating and producing knowledge. *Slash Goggles* inhabits this space in order to proceduralize sexual identification, exposing the ways in which one is not without the other, if we care to trace the performance of gender--and other categories--systemically across digital media. Through a conglomeration of rhetorical code practices and performative investigations, scholars will begin to see how seemingly disparate texts, materials, or technologies actually co-constitute one another. The critical is more than ever experimental, in the place where experimentation seems the most constrained.

This essay demonstrates how computational performativity is a gender performativity. This is not simply the production of effects in machines and users, but the production of subject positions tied to gendered epistemological practices. Both Russo and Blas respond to the rhetoric of software engineering, the how of code circulation and performance, to comment on topics of gender construction, sexual identity, and how both play out in digital technology. But, as I have argued, this critique expands into a queer technique of rhetorical engagement with code that recognizes how structure, form, data,

and complexity shape and constrain discursive positions in coding as a discipline. Computational performativity, in the manner I have discussed, offers the potential to look for gendered practices across computer code--not as representations of gender through discourse, but as the ordering of media and media consumption as gendered to produce discourse. While I believe there are numerous implications for the application of such analysis, the primary vision of this argument is to offer an approach that recognizes code as epistemology, epistemology as gendered and performative, and as such the potential to see a continuum of gendered practices across software and its use. Such a vantage, I believe, offers two immediate benefits.

First, scholars of a critically queer code studies can develop a body of work that explores the relationships between gender representations in STEM industries and academia through software--its production, maintenance, and circulation. Because performativity, computational and gendered, is the continued creation of new worlds, a focus on the queering of software at its own game is a critical imperative. Second, as studies in the fields of computer science, computer engineering, and electrical engineering have already begun to note the skewed representation of cis-gendered masculinity in programming, a productive interdisciplinary framework between STEM and studies in queer sexualities can emerge from a shared critical discourse. An increasing number of studies from the fields of computer science and software engineering point towards gender and racial disparities, not just evident in hiring patterns but in how these fields unknowingly invite or exclude certain individuals through their cultural assumptions and teaching and working practices. While questions of literacy, education, and social attitudes towards gender and engineering all play into these

conversations, critical gender scholars can collaborate with these fields to further investigate how these social realities are present and productive in software. Collaboration in this key stands not only as a corrective measure, but as a way to inform how computer scientists produce software, digital subjectivities, and with them our contemporary culture.

Chapter 3
BitTorrent, Swarms, and Collaboration

Introduction

Digital technologies structure how we work with one another. While this has immediate and intuitive ramifications for those who use technology, it takes on a different scope and scale when applied to those who produce this technology. Free software pioneer Richard Stallman wrote in the "The GNU Manifesto" (1985) that "[s]oftware sellers want to divide users and conquer them," referring to the way software companies sought to impose traditional licensing and copyright models to source code. What makes his manifesto significant, as Stallman himself points out, is that "users" are also the "producers" (i.e., programmers), and produce by sharing information like code that would normally remain proprietary. The conceptual collapse of the user and the producer raised (for Stallman) key questions about how access to information structured collaborative work. Building software was not like building a car—it is not enough that everyone does his or her job, but that everyone understands how their work contributes to a larger process of programming. Along these lines, we must admit that large, production-grade software rarely comes to fruition through the hard work of a single laborer, but rather through teams of engineers, communicators, and managers that negotiate and interpret the vision of a project through a cultural feedback loop that runs through different contexts of expertise, enactment, and exclusion or inclusion.

The figure of the lone, brilliant programmer persists in many areas of software development, primarily emerging from stereotypes connected to academic computer science and representations of awkward, often socially maladjusted technical specialists who dwell in the bowels of some server room to emerge to help with troubleshooting or maintenance. The previous chapter articulated the ways in which mastery functions within a gendered framework of labor rooted in a professional history where mastery is made synonymous with masculinity to the exclusion of a clerical "feminine" workforce. Alongside this, the notion that programming, software engineering, or computer administration was ostensibly a masculine pursuit continued into professional and academic contexts for decades.

I believe that there is an ironic relationship between masculinized technology and the question of collaboration. This chapter continues this line of thought by examining the relationship between technological objects and how they express modes of collaboration within that same industry, and how those modes correlate to gendered assumptions of collaborative labor and communication. My primary area of examination is the file sharing protocol BitTorrent. Popular understanding of BitTorrent is a bit limited, in part because it has a certain mystique as something targeted for those who are more technologically savvy. By looking at the logical structure of BitTorrent, however, I see the relationship between the protocol's articulation of "swarm" collaboration and Connell and Wood's (2005) analysis of transnational business logics, and more specifically the ways in which certain forms of collaboration and organization become gendered and subsequently valued or devalued. This illustrates the assumptions innate to

technical objects like protocols, and links such expression to the tension of agency in networks more broadly.

As I investigated in Chapter 2, the relationship between digital labor, collaboration, and gender are not only a question of representation in the field of computer programming, but also a question of the labor itself, insomuch as techniques in computer programming can reflect gender relationships. Primary to this was an expression of the notion of the ideal, singular master of code as a discourse. Following this, I also argue that foregrounding criticism of code that focuses on individual pieces of code (like the algorithm or the interface) may implicitly reinscribe or valorize that frame of reference for code. Proceeding from my analysis of the performativity of code in Slash Goggles and transCoder, I now look at the question of the "protocol" to extend my analysis into the question of relational logics of labor more broadly, maintaining my focus on their expression digital artifacts. Protocols are not simply rules of connection or engagement, but blueprints for connectivity that prescribe specific styles of network formation. The cultural work of a protocol is a demonstration of the collaboration across the boundary of user and expert, which exposes the gendered histories of collaboration and labor. Rather than see these histories as encapsulated in a protocol, however, I argue that they are rather distilled or refined into concrete logics of collaboration. Thus, technical specifications can be read as a cultural expression of ideology much like any other text.

Drawing from my arguments in the previous chapter, I further contend that protocological expression can and does extend beyond texts like computer code, and find further implications in what we would colloquially refer to as "networks." However,

rather than spreading that terminology to its breaking point, I will limit my discussion of networks to technological or communicative networks grounded in digital media. There are connections between the strictly technological network I discuss here and "networks" more broadly, but one specific difference, other than a limit of scope, is a specificity of application. Technological networks are defined by function and form, providing efficient and reliable modes of communication. They are also, as Alexander Galloway (2004) writes, prescriptions of structure and power that frame modes of communication (54). In their explicit call to the relational aspect of communicative labor, protocols more accurately gather hardware, software, and users within a logic that defines ways of collaborating that are also inflected by histories of labor and gender in digital technology. Or, as I suggest, protocols articulate certain rhetorical formations. I draw the term, "articulation" from communication and cultural studies to describe the expressive work of protocol. Articulation is tied to the practices of identity in technical communication in that meaning, communication, and mediation communication networks are better realized culturally as a series of communicative identities that potentially re-inscribe or resist dominant cultural ideologies. More specifically than that, through articulation and the notion of performativity, I will argue here that "articulations" of identity are present in hardware, software, and users through myriad computational utterances that perform arrangements of collaboration that have been historically valued with gender characteristics.

BitTorrent

Despite its technological mystique, BitTorrent represents an attempt to reinterpret one of the most fundamental operations in digital communication: the transference of files between one user and the next. The goal of the protocol itself, as outlined by creator Bram Cohen (2001), is to improve the efficiency and reliability of file transfer for larger files through a collectivization of downloading and uploading resources. Cohen presented the specification and tested it over the next two years before developing and releasing the software. Protocol specifications, seen at a glance as impenetrable and iron-clad design documents, serve as connecting definitions of logic that specialists interpret and enact in hardware and software.

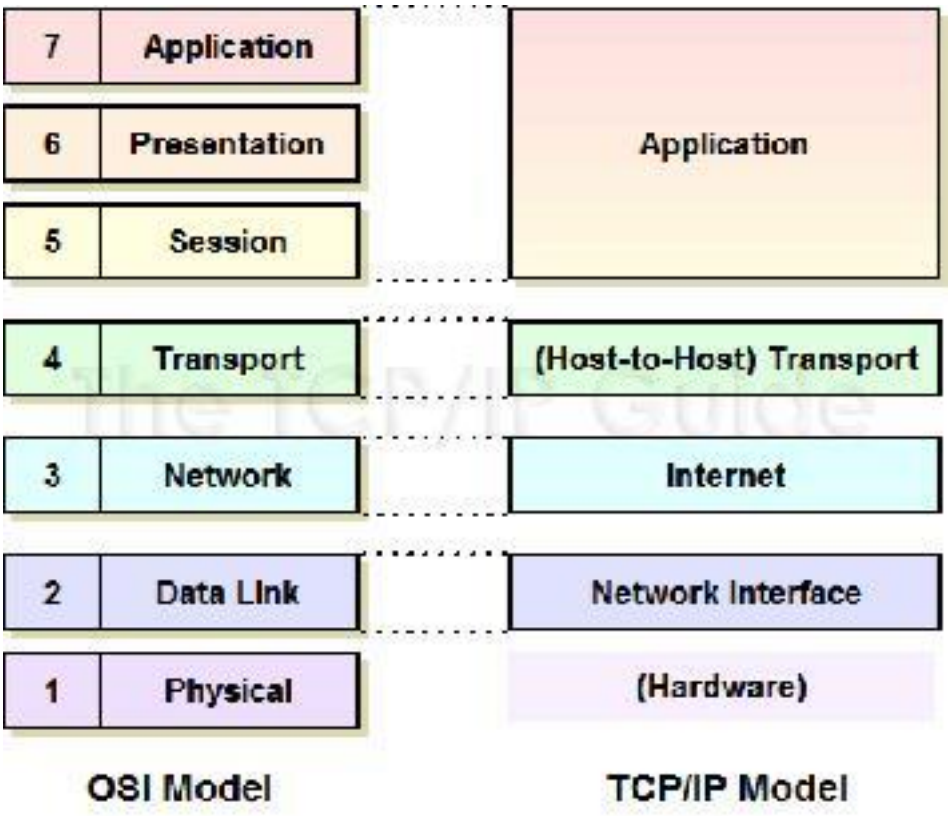


Figure 3.1 The TCP/IP Stack

The BitTorrent specification is no different, but if we recognize its highly technical nature as a performative aspect of a larger technical/cultural identity, it gestures to some of the larger cultural assumptions. To shape the contours of this performativity, it is necessary that we first understand the technological context of BitTorrent as a piece of software and then approach how it functions at the level of critique. A discussionof the BitTorrent protocol provides a way to investigate the intersections of the technological, performative, and rhetorical dimensions of BitTorrent as a form of organization and critique. The simplest way to discuss a protocol in the digital sense is to discuss it in tandem with another concept: the network. And the simplest way to discuss a network is to discuss the one that we are most familiar with: The Internet.

TCP/IP and the Internet

A quick etymology of the name "Internet" demonstrates that while a network, it is more accurately described as a network of networks (inter-net) and, as such, a specification by which decentralized clusters of computers can connect into a larger, cohesive communication structure. The Internet does not, despite some remaining popular belief, itself exist as a decentralized collection of computers, phones, routers, and cables that have come together and emerged as some sort of cohesive structure. As Galloway writes, were this the case the Internet would fail as a meaningful mode of organization or structure (30). Nor is it a metaphor for our contemporary condition, a flattening of the global economy and social connectivity. Rather, the Internet is the result of a structured suite of rules, definitions, and operations that defines how those computers, phones, and routers communicate through those cables.

The structure of the Internet is defined by the "Transmission Control Protocol/Internet Protocol" (TCP/IP). The first development of TCP/IP is credited to Robert E. Kahn and Vincent Cerf (1974) as part of the Defense Advanced Research Projects Agency (DARPA) project on computer communication, which sought to create a robust and flexible network of computers for military application. Kahn, Cerf, and their team (in conjunction with the United States Military and several universities) sought to create a network that did not have to rely on a central host or controller to function, and maintained itself through a distributed operational model. This is likened by Galloway to President Eisenhower's plan for the national highway system: a plan where travel or exchange can continue even if massive damage occurred to any given point or points (35). As such, there is no "TCP/IP server" or enabler that shapes the Internet (this is left to other technologies, as we will discuss later). Instead, TCP/IP constructs a self-producing network in which nodes, or points within that network, share the responsibility of routing communications and keeping the network up and running. To accomplish this, the TCP/IP protocol was conceived as an approach by which communications would occur through the combined efforts of the machines on that network, and that the operation of the network would be independent of software used by those machines. This, in turn, required a clear specification of the rules by which computers would communicate, and a method by which different computers could utilize the same underlying rules. W. Richard Stevens (1994) writes that "[n]etworking protocols are normally developed in layers, with each layer responsible for a different facet of communications" (5). These "layers" are typically called the TCP/IP "stack," represented in a stack diagram:

The "stack" is a theoretical representation of how data moves through various hardware and software to find its way to its intended destination, and serves as the extensible model of how TCP/IP works. The "application" layer is the layer we most readily recognize. This layer is where we interface the data, either through a program or through the operating system. A web browser is a good example, as a web page served and visible through the web browser is at the application layer. The data was meant to be at that layer, and interpreted by that software. The next level down, the "transport" layer, breaks the data into pieces, called "packets," and includes information on how many packets there are, the size of the data, and how it will connect to the server. The third, "Internet," Adds information regarding the location of the receiving computer on the Internet, such as the IP address, and the address of where the information came from. This is critical for error checking, as an error in transmission can be served to the sender. Finally, the "Link" layer adds a "MAC", or hardware address from the sender and receiver, so that the proper hardware (typically an Ethernet or wireless card) get it.

The significance of this is that the stack represents an order that abstracts the reality of the actual implementing computer and, eventually, the material substrate of communication, at the bottom of all this data. Even the name “TCP/IP” denotes this reality: Internet Protocol defines the basic kinds of information passed around between computers (in this case, data quanta known as “packets” and strings of numbers called “addresses” that denote location) while TCP defines controls that render this information and its transmission reliable. Because of the shared simplicity of the information quanta and the abstract approach of the controlling mechanism, each layer, while working in conjunction, also defines an entirely different context through which information passes.

This allows programmers to develop applications (or other protocols) at different points in the stack without necessarily having to consider another layer (except, perhaps, to know how to handle data). Whether a developer uses a Mac or PC, Windows or Linux, Intel or AMD hardware, the implementation of the protocol allows all these computers to share a common mode of communication. Independently of how I am utilizing the data (browsing the web, transferring files), the TCP/IP stack facilitates the communication by abstracting network communication from the user.

However, TCP/IP is only part of what makes the Internet the Internet. TCP/IP, by itself, is a radically horizontal and dispersed logic of control, and furthermore it uses conventions that are not necessarily helpful to users: for example, IP addresses uses to designate computers are complex and hard to remember, and do not provide an intuitive organization structure from end-to-end. This is where another part of the Internet comes in, and renders it in many ways intelligible to the everyday user. Dynamic Name Server (DNS) protocol, exerts a different kind of control that modifies how the network emerges. DNS defines the method of storing, distributing, and decentralizing a list of Universal Resource Locator (URL) names mapped to IP addresses across the Internet. So, for example, if the IP address for the Google main search page server is 74.125.29.99, then the DNS protocol specifies the way in which computers routing IP packet can resolve the IP address (74.125.29.99) with the URL name (google.com). DNS, unlike TCP/IP, is hierarchical in the way it organizes IP addresses, as the following diagram illustrates:

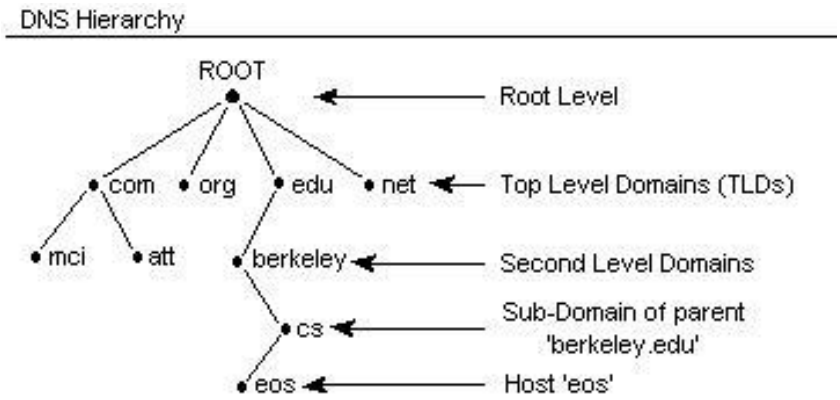


Figure 3.2 DNS Model

The overlap of TCP/IP and DNS therefore creates a meaningful logic of communication, and from that emerges a network in which routing, tracing, and domain-name lookup are distributed throughout the nodes.

Rather than the hierarchical nature of DNS, or the decentralization of TCP/IP, the Internet operates through what Galloway calls a "contradiction of two opposing machines," which belies the reality that the Internet, although appearing decentralized, is tightly controlled (9). That is, the non-linear TCP/IP protocol works alongside the structure of DNS to form a robust and usable network. To look at the Internet as a protocol, rather than as an emergent assemblage, requires we study the protocol specification as a precursor to network formation.

As Panwar, et al (2004) write, "The lowercase Internet means multiple networks connected, using a common protocol suite. The uppercase Internet refers to the collection of hosts . . . around the world that can communicate with each other using TCP/IP" (16). Thus TCP/ IP abstracts what Tiziana Terranova (2004) would call the "heterogeneous" collection of local networks, producing a system of mediation in which they all therefore become a larger network under the same logic of communication (140).

The strength that comes from these two seemingly opposed logics is that they ground responsibility for network maintenance into a repetitive set of actions and data structures shared by all computers. IP packets move through the network through a process called “routing,” where a series of computers will receive packets from one end of the network and, utilizing the IP address of the sender and the receiver, “route” the packet to the next nearest computer for further transmission. Data is rarely, if ever, transmitted directly between two machines, but along a route of connecting machines that direct traffic where it needs to go:

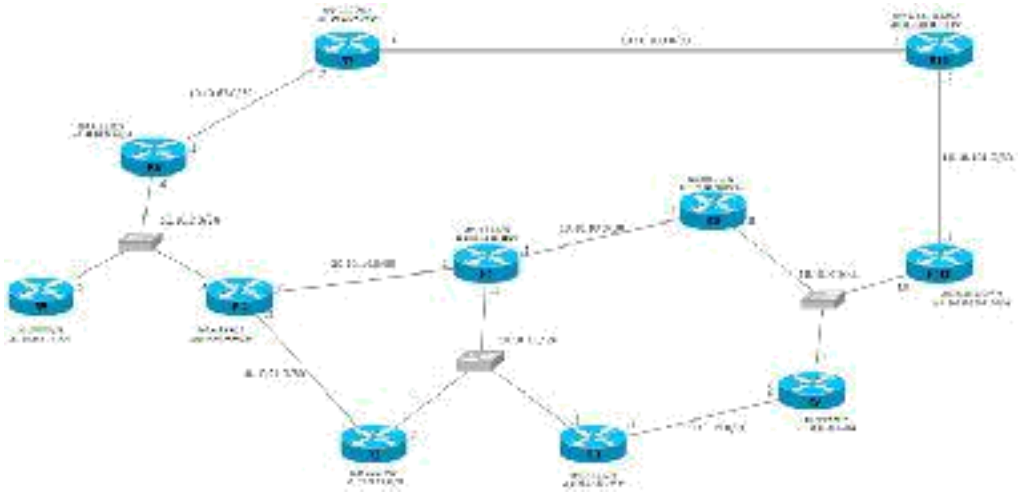


Figure 3.3 Routing Table

As much as we think of networks as participatory (I "get on" the Internet), protocols define the ways in which each user is harnessed as a propagator of that network. The Internet is no exception: the basic logic of TCP/IP is a robust persistence, an iteration across implementing machines. Each node in the TCP/IP network is a shared production of the protocol, and each node, although ostensibly representing a "user" (and this is the point where we might easily conflate a node with an actual entity, like a

computer or a human), is a point of the network's production of itself. As Galloway writes,

> Thus, in an important way, networks are not metaphors. The network metaphor is misleading, limiting. It only provides a general model for discussing relationships. . . Understanding networks not as metaphors, but as materialized and materializing media, is an important step toward diversifying and complexifying our understanding of power relationships in control societies. (xv)

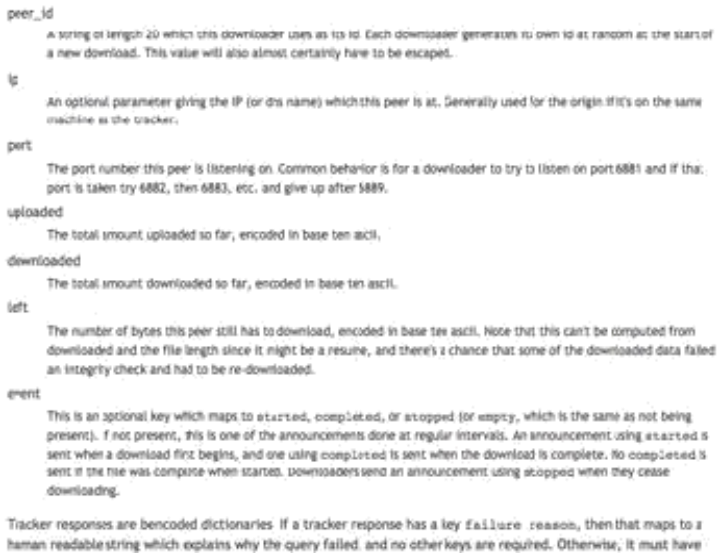
If networks, and network protocols, are not metaphors, a pivotal rhetorical mechanism of the network is not necessarily the communication that moves through it, nor necessarily the potential that it enables for users, but its own reproduction of itself as an expression of relationships.

BitTorrent

Judy Wajcman (2004) argues that the gendered and technological aspects of networks are already social and technical, constitutive social and material realities (104). In that sense, the realities introduced by a network protocol, are concurrently social and technical, and not simply in their most socially recognizable manifestations. The specifications themselves, the documentation and the definitions, constitute a clear social construct that illustrates the interconnected work of communicative action in technology. They also outline the spaces in which awareness of social action, engagement with it, and (potentially) critique of it is merged into a single socio-technical space. We saw this very clearly in Chapter 2, where code and sexuality mingled into a space of discursive potential rooted in digital media and history. That reality continues here, writ much larger and spread across a wider spectrum of hardware and relationships.

Take the application layer of TCP/IP. At this layer, we find numerous attempts to facilitate new kinds of communication, to harness or create new networks "on top of" the TCP/IP protocol, to accomplish different tasks. One basic function that happens at this level (and often in conjunction with other levels) is the process of file exchange. HTTP and FTP, for example, are traditional file sharing protocols that we typically use or have used daily to exchange data in the form of web pages or documents. Traditionally, file sharing occurs through a straightforward client/server model, in which files are hosted on centralized servers. To get the file, a client computer connects to the server, downloads the file in its

entirety, and disconnects. A single server, hosting a single file, would serve that file to as many users as connections could be maintained, but for each user, the file would be downloaded in its entirety from that single server. This model obviously causes problems when servers are overloaded by too many users, or if connection to the server is unreliable.



peer_id

A string of length 20 which this downloader uses as its id. Each downloader generates its own id at random at the start of a new download. This value will also almost certainly have to be escaped.

ip

An optional parameter giving the IP (or dns name) which this peer is at. Generally used for the origin if it's on the same machine as the tracker.

port

The port number this peer is listening on. Common behavior is for a downloader to try to listen on port 6881 and if that port is taken try 6882, then 6883, etc. and give up after 6889.

uploaded

The total amount uploaded so far, encoded in base ten ascii.

downloaded

The total amount downloaded so far, encoded in base ten ascii.

left

The number of bytes this peer still has to download, encoded in base ten ascii. Note that this can't be computed from downloaded and the file length since it might be a resume, and there's a chance that some of the downloaded data failed an integrity check and had to be re-downloaded.

event

This is an optional key which maps to `started`, `completed`, or `stopped` (or `empty`, which is the same as not being present). If not present, this is one of the announcements done at regular intervals. An announcement using `started` is sent when a download first begins, and one using `completed` is sent when the download is complete. No `completed` is sent if the file was complete when started. Downloaders send an announcement using `stopped` when they cease downloading.

Tracker responses are bencoded dictionaries. If a tracker response has a key `failure reason`, then that maps to a human readable string which explains why the query failed, and no other keys are required. Otherwise, it must have

Figure 3.4 BitTorrent

BitTorrent changes this by shifting peer-to-peer exchange from a client/server model to a decentralized client/servers model that offloads the work of transmission into a crowd of users. A "swarm," in this case, refers to the way multiple users, of their own volition, participate in a set of shared activities to distribute work and computational resources to accomplish some task. When downloading a file from a source (say, a web page, or an audio file, or streaming data from YouTube), the relationship from client to server is one-to-one: the client connects to a single server and downloads the data. Conversely, a server to client relationship is one-to-many: a server can deliver the file for many concurrent client computers, depending on resources. This is because a file, for the purposes of transfer, is a single entity clearly demarcated by the file systems in question and bounded by meta-information that tells each computer where the file begins and ends. I download

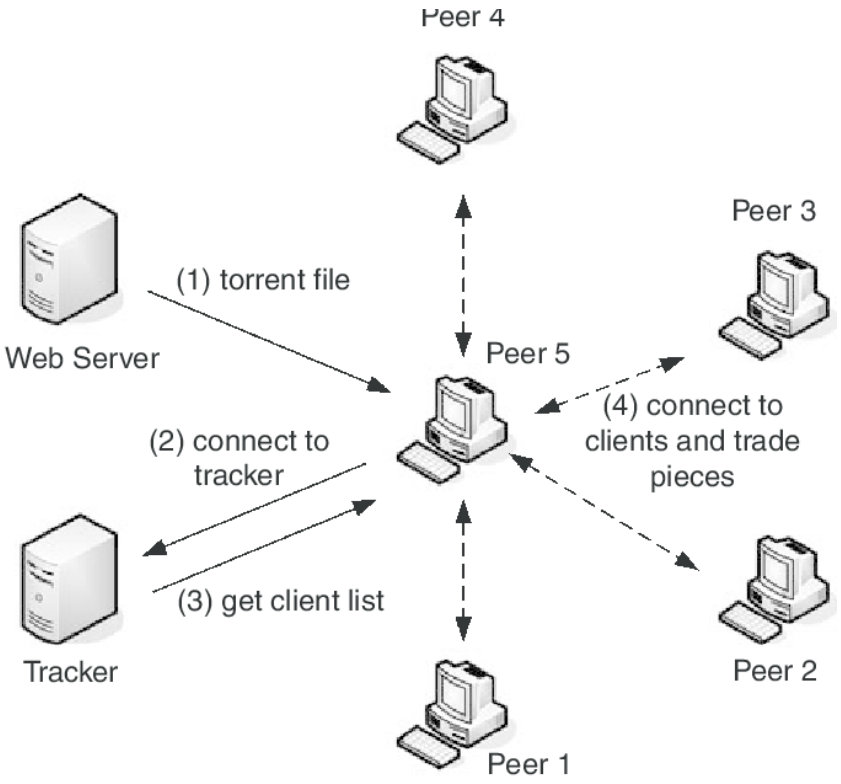


Figure 3.5 Traditional vs. Swarm Servers

a music file, and I get, from that server, that musical file. It is a single, contiguous entity. Under BitTorrent, however, each file is represented as a segmented file:

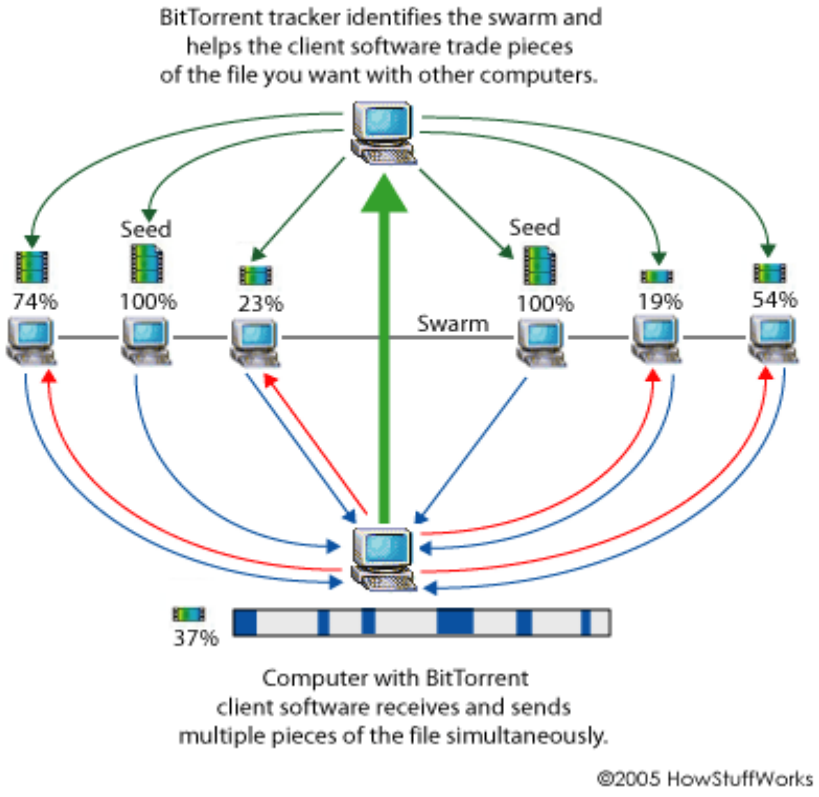


Figure 3.5 Traditional vs. Swarm Servers

That is, each file is divided into segments called pieces which serve as quanta for file transfer. An associated torrent file spells out the metadata for the file of transfer, including the size and number of pieces. Trackers, or programs that monitor these torrent files, trace the number of users with the torrent file and the original data file, and keep count of how many computers contain that file. Following this, when a client downloads the torrent file from a server, the tracker uses the file to connect that user to a swarm of computers sharing that file (Cohen 2003). This is significant, because instead of modeling a direct client/server relationship, BitTorrent models multiple client to server relations to

access data. On its own, this fact doesn't really mean anything, as a computer can connect to servers as much as it please. But BitTorrent changes this by structuring the communicative relationship between computers so that each connection my computer makes to a server is a download for a portion of the file, rather than for an individual file.

To maintain the robustness and efficiency of this network, each user or node on the network thus falls into one of two categories. First, "seeders" represent nodes that share a file. Under the traditional mode of file sharing, a node fulfills the "server" role when it makes a file available for copy over networked communication, and other computers download from it. Seeders are computers that serve the files, but under the rules of BitTorrent, each seeder is responsible for sharing only pieces of that file, rather than having to share the entire file (Cohen 2003). Second, "leechers" are nodes downloading the file pieces from other computers. A leecher, downloading a file, foregoes the trouble of having to download an entire file from a single server, but instead downloads pieces of that file from multiple seeders concurrently. This means that, depending on the ratio of seeders to leechers, a client could still download an entire file from a single source, or download multiple pieces of the file from multiple computers (Cohen 2003). Downloading pieces from multiple seeders dramatically (again, depending on the ratio of seeders to leechers) increases the speed and reliability of the download while decreasing computational load and connection times. A collection of seeders and leechers exchanging pieces of the same file is called, in BitTorrent parlance, a "swarm." The protocol effectively distributes responsibility through time and space to make the network work, and this distribution creates the swarm of users that collect around a given file, each fulfilling their role. BitTorrent therefore produces informational "bodies"

within a network that complete a very specific and discrete task that follows a swarm logic (Cohen 2003).

Protocol and Critique

It is easy to see how one can get lost in the technical discussions related to software and lose sight of its immediate cultural impact, which is why it is often easier to see the cultural impact as limited to the use of technology rather than the technology itself. I included a selection from the BitTorrent specification above to demonstrate this, and to demonstrate the translation work required to begin to work through the larger implications of the protocol itself to see it as structurally expressive. This expressiveness is in fact a critique of inefficient file transfer and a rectification of that through an extension and rethinking of that underlying structure, which then in turn expresses a swarm network. The BitTorrent protocol specifically uses the same contradiction of collective and individual agency in TCP/IP, but mobilizes it differently to produce the network shape that it seeks.

"Structurally expressive" means different things in different contexts, and through different technologies. One might ask what BitTorrent expresses, and in doing so we enter a different realm of analysis that breaks away from the notion of technology as an accompaniment to, or mediator of, effective rhetorical action. Brown (2016) argues that assumptions of ethics are "coded" into software, and upon looking at BitTorrent we immediately notice some of the cultural work of collaborative and networked labor expressed. For Brown, this invites us to think of the "possibility spaces" in code, or to think about how code provides a new space for rhetorical action (99). However, I do not

want to get too far into a strictly rhetorical discussion of the expressiveness of a given technological objects, in no small part because I believe that it may in fact re-inscribe the glorification of individual utterance or discourse mentioned earlier. For example, Byron Hawk (2004) articulates a vision of rhetorical theory in which common topoi like ethos, Heuristics, and Situation find their re-inscription in an informational communicative discourse (832). Likewise, Collin Brooke's (2009) Lingua Fracta re-situates the rhetorical canons considering a new media culture, where comment sections, databases, and the Internet place each canon in a slightly-askew position, a reference to something similar and yet different (199). As Brooke writes, however, his study of new media and rhetoric is more about how communities "take up" technology, and Hawk's theoretical approach likewise reconfigures existing rhetorical tropes within the taking up of complexity theory and digital media. Brooke specifically cites the fact that he focuses on the level of "practice" rather than code or culture to frame his work as a more generalized heuristic for thinking about rhetoric and new media (199). This practice seems to take root, however, less in the collective power of a given technology and more on the potential for persuasive action within technology (as opposed to outside or prior to it).

As I consider code as performative of historical norms of labor, I do so with the understanding that code and protocols are in fact utterances with ramifications for further action. That is, performative technology is something that is done rather than something that is external to the doing. BitTorrent, for example, is in some sense a constraint upon communication. But a performative understanding of it as an act sees the critical capacity therein to resist, subvert, or comment upon other performative technologies. BitTorrent harnesses the "doing" of TCP/IP and DNS (the structuring of relationships in a public

network) to do something else, something that extends from the protocol that came before it. I therefore resist then the urge to make technology a discussion of place, of where discourse happens or where speakers and writers make persuasive texts. Robert Kitchen and Martin Dodge (2011) frame the studies of code and network as one of space, arguing that code or network space is produced by the production of assembled actants that "emerge as self-organizing systems of relations stretched out across space and time" (77). Case in point, many approaches do not necessarily address what Donna Haraway (1991) refers to as the "control strategies" that exist in the technological societies of today. She writes that "one should expect control strategies to focus on boundary conditions and interfaces, on rates of flow across boundaries . . . No objects, spaces or boundaries are sacred in themselves; any component can be interfaced with any other if the proper standard, the proper code, can be constructed for processing signals in common language (302).

Boundary conditions, unlike spatial boundaries, are the meeting points of practices and control, of a self and the production of that self, in informational society. More specifically, these interfaces are the strategies of controlling disparate, heterogeneous entities through the re-production of those entities within an informational context. So, computers developed across different manufacturers with different specifications are recreated as homogeneous nodes through a series of regulations and specifications. The exercise of any agency is then already linked through this reproduction, with the performative work of coding structuring the exercise of agency in a tension between individual activity and collective necessity (302). Women and reproduction, Haraway argues, are (re)figured in discourses of technoscience as problems

of population control and "maximization of goal achievement for individual decision-makers" (302).

It is precisely the seemingly reasonable and accepted relationship between technology and culture exemplified in "network culture" that makes control strategies not only possible and potent, but normalized into previous frameworks of inquiry like rhetoric. Terranova defines network culture as an informational milieu in which the characteristics of data, communication, and non-linearity shape social-technological formations. Concepts like social networking and the citizen-content producer were still in their infancy near the beginning of what is now commonly referred to as the "Web 2.0" of wikis and web applications that we are familiar with today. But her arguments regarding the nature of information, the Internet, and "free labor" resonate perhaps even more strongly now than ten years ago, primarily because the consequences outlined in Network Cultures are only amplified through contemporary, participatory social media. When she argues that "the relation between the Internet and the production of space is . . . crucial to all theoretical engagements with Internet culture," it is important to understand that she does so to note that "this informational and electronic space, as it is constituted within this single map, appears as uncannily reminiscent of a modern dream for a completely homogeneous and controllable space" (44).

This seems counterintuitive: what could be more heterogeneous than the Internet? And yet even in the discussions of TCP/IP and BitTorrent we can see some of the truth of this, and further see the way in which these protocols express homogeneity. In "Articulation: A Working Paper on Rhetoric and Taxis," Nathan Stormer (2004) argues towards a rethinking of rhetorical history and theory as one "that does not require

principle focus on a humanistic subject, centered or decentered" (258). For Stormer, this understanding places the study of rhetoric as the study of practices, and the study of practices as the study of relationships between entities rather than the entities themselves. Unlike Brooke's claim to study practices, Stormer argues that studying practices is the study of relationships and order itself, and how order expresses particular rhetorics. He writes that "as opposed to analyzing the rhetoric of individuals, analyzing the rhetoric of practices means one does not necessarily start from a given point nor end in each place. One may start with a space, an image, a canonized activity, or a network of relationships, and study their importance to the formation of the culture" (259). Karen Barad's (2008) theory of material performativity resonates with this understanding of rhetoric as emergent. She writes that "discourse is not what is said; it is that which constrains and enables what can be said," and clearly ties constraint to material conditions outside of human language (261). For Barad, the performativity of discourse is found in the dynamic interaction of the world, and calls for prioritizing relations over pre-existing objects entering in to relation. She argues that the limits of Butler's theory of performativity is that it does not account for the performative materiality of bodies themselves, and in conjunction with Stormer I would use the term "body" as a reference not only to human bodies but to rhetorical or informational bodies, bodies rendered by technology and code.

Articulation in this technological sense is the activity of performative technology as it expresses particular cultural ideologies through media. Obviously, the term also has an established history in cultural studies, but has also made its way into scholarship in technical communication. Slack, et al (2006) draw from Alan Grossberg and Antonio

Gramsci to integrate articulation as a concept into technical communication. To understand articulation is to understand that "any identity in a social formation must be understood as a non-necessary connection between the elements that constitute it" (37). It is not much of a stretch to then understand network identity as something constituted by the protocols and relationships within that network, and to then look to those protocols and relationships as significant rhetorical aspects of that network. To follow this line of reasoning also means that the continuum of user-technology-expert is only a particular formation of power that designates a spatial and temporal framework to understand experts as creators of networks, users as inhabitants of networks, and technology as metaphorical or literal spaces, sites, or locations. With each separation, with each dash, we immediately render the distinction between each clear, and in particular allow analysis to exclude specifics of cultural ideology from seemingly non-cultural aspects (that is, technology).

BitTorrent, perhaps surprisingly, demonstrates the relationship of control and articulation to digital media, and immediately problematizes the separation of an agent or actor in a network and the technology that constitutes that network. Leechers and seeders are not simply labels placed upon users as they participate in the technology: they are conditions of the technology and its performative utterance. The designation of actors as collaborative aspects of the network articulates a worldview of collaboration that values distinct separation of duties, normalized relationships that inform a larger goal. Nodes—seeders and leeches—are further articulations of the protocol, and the protocol itself is re-articulated through use, error checking, and testing. Likewise, the tension of individual agency and the construction of agency for collective purposes highlights the critical

importance of recognizing the ways in which these ideologies permeate technology. Brown rightfully argues that networked environments "make it difficult to hold situated [socially produced or oriented] ethos and invented [constructed] ethos," as distinct practices, but limits his discussion of how ideological categories such as race, gender, and class play a part in either in these environments (Brown 113-114).

> BitTorrent becomes problematic when it is theorized as prior to or differentiated from

ideologies of power, or when it is used to discuss rhetorical concepts like invention, distribution, or circulation outside of such ideologies. Brooke writes that "I do not question that factors such as gender, race, and class have profound effects on technology, but I would dispute the notion that technology studies should be reduced to an exploration of those factors to the exclusion of all else" (Brooke 13). I don't believe that his point here is to disparage gender studies, but rather to argue that there is a technological discussion of rhetoric outside of them. The interesting use of the word "reduction," however, suggests technology as differentiated or more primary to the study of gender (or perhaps even any other social category), a notion that I reject. As Theresa de Lauretis (1987) writes, "the movement in and out of gender as ideological representation . . . is a movement back and forth between the representation of gender (in its male-centered frame of reference) and what that representation leaves out, or, more pointedly, makes unrepresentable" (26). Technology is often the space where gender is made unrepresentable, and too much emphasis on the "language" of technology marginalizes the clear cultural affect of that technology.

BitTorrent, as an expressive discourse, is a sort of apparatus that produces informational bodies. BitTorrent produces these bodies through the performative nature of digital media: the epistemological underpinning of TCP/IP extends through BitTorrent as it multiplies the server-client model to produce a swarm, a force that disassembles and solves a problem. In the BitTorrent network, the "nodes" are leechers or seeders, but more accurately they are composites of file transference. That is, each node in the network is ostensibly, within the parameters of participation, an agent of the swarm or the shared responsibility of file transfer. They hold pieces of files and allow access to them, while they also download other files. Thus, the central tenet of BitTorrent is that the more participants engage within the process of sharing or swarming, the more stable and productive the network is. And, the more effective its tactics are: many a message board has bemoaned the abuse of the system by "leechers" who refuse to seed (participate). Contradictory logics in protocols have been referred to as "ironic" specifically to point out how seemingly these contradictory elements, rather than functioning in a deconstructive framework, instead articulate the discrepancy between network protocol as models and networks as performative entities. TCP/IP functions through the enactment of a hierarchical and a decentralized juxtaposition, a mode that allows it to distribute an agential network in which all computers can feasibly run as servers or clients. Likewise, BitTorrent, in building off TCP/IP, draws from that performativity to articulate a swarm network, adopting and multiplying the server-client relationship. This expression of multiplication results in the swarm. BitTorrent uses the term "swarm" to refer to the work done, continuously, to share and maintain the sharing of a file (Cohen). Inspired by Haraway's Cyborg, Sarah Kember (2003) argues that Haraway's study of

biotechnological kinships are not "purely epistemological," but they are a "response to different modes of being emerging from the mutually inflected fields of biotechnology, feminism, and science studies" (187). This conversation points to what Barad calls the "agential cut," where the logic of relationships define agents, rather than the other way around (Barad 144). There is no direct distinction of technology, biology, or language prior to such cuts, but instead the production of bodies through material-discursive practices. BitTorrent is an enactment of and formation for swarm logics, and participates in the production of the informational bodies necessary to produce agency. However, the "cut" of BitTorrent emerges not from a space of equanimity or horizontal organization, but (like other protocols such as the TCP/ IP protocol that it builds upon) a goal-driven and hierarchical logic that, in turn, articulates a seemingly decentralized plane of shared labor towards a common goal: the potential for communication and participation. The swarm logic of a protocol like BitTorrent does not, despite claims to the contrary, produce an emergent whole from disparate parts, but rather a controlled collapse of individual and collective agency under the ironic juxtaposition of control logics in technology.

The Periperformativity of Protocological Critique

If BitTorrent as a network articulates informational bodies from its specification, then it is structurally expressive. It is also, in its own way, a critique of the underlying file-sharing technology that came before it. Countering the inefficiency of the client-server model, BitTorrent enacts its critique by articulating new forms of collaboration. The nodes come after the logic of these forms, and the information that passes between nodes

is not the product of pre-existing agents, but a performative utterance of a network articulating itself. If abstraction has its roots as a historically gendered technique of collaboration, then abstraction as a practice also allows different historically gendered techniques to emerge as consequences of actions undertaken through software. BitTorrent distributed computer power to share files, but swarms in other contexts distribute labor that has traditionally been gendered in the technological world (writing, transcription, data entry) and hearkens back to a history in which gender and sexuality functioned as delimiters for acceptable work while at the same time constructing a system of invisible work.

Since BitTorrent is an articulation of agents within a collaborative network, a rhetorical understanding of that articulation should engage the politics of that collaboration. As Karen Lunsford and Lisa Ede (1990) argue, there are multiple modes of collaboration or participation, all of which can accomplish some goal for some reason. Some differentiate a "dialogic" mode of collaboration with a "hierarchical" mode. The hierarchical mode is driven by strictly delineated tasks accomplished by individuals in clearly assigned roles (235). This linear mode of collaboration values the individual as part of a greater whole, with the greater whole structured as a clearly defined task or goal. This form of collaboration is less interested in a multitude of perspectives, and as such it often seeks to conform them into a single goal or purpose to smooth out the collaborative process. Lunsford and Ede argue that this mode of collaboration is typically conservative and historically inscribed as masculine, and often sees collaborative or participatory work as a problem to be solved (235). These definitions are not universal, but remain reliant on different kinds of institutional or cultural labor. Connell and Wood (2005) write that

patterns of management (patterns that structure worker relations for subordinates, among other things) often demonstrate forms of hegemonic masculinities that reflect changing modes of collaboration (349). They write that masculinity in global business contexts often reflect progressive attitudes towards gender equity while at the same time reiterating highly stressful and limiting gender traits seen as necessary for business people: namely the exercise of power in institutional relationships and recognition of provisionality as part of that power (360). The separation of progressive representation of gender equality from gendered labor practices allows for the rise of business masculinities that value competition, power, hierarchy, and provisional social relationships.

Such a separation is evident in many professional programming communities, perhaps magnified through their fluid nature and overt focus on technological production. Linus Torvalds, the creator of the Linux operating system, has often equated masculinity with technical work like programming, often connecting complex, mathematical tasks and design to the work of "men." And while there is often a strain of gender discrimination in programming more broadly, much more significant is the ways in which increased calls for progressive reform in programming culture continue apace with the very same styles of collaboration that mimic hegemonic business masculinities. Linux is grounded in the notion of global, provisional labor, where power is modeled through meritocracy, and where competition (good code) is seen as the ultimate success and proof of value. The styles of collaboration, which seem to foreground collective work and communication, do so through goal-driven and provisional models of participation that

do not necessarily value concrete relationships outside of professional or success-oriented models.

In bringing up the question of collaboration, I hope to tie together the threads of this inquiry into a more coherent example of performative utterance in the form of the protocol, and to emphasize how types of protocological utterances affect network formations and, in a sense, further articulations. Network protocols map out the relational structure of a network, and as such they further articulate that network through key values. This is significant for both rhetoric and studies of gender in that the power to define relationships not only structures rhetoric or rhetorical possibility, but also the very conditions of agency in communication technologies. Stormer writes that "for me, articulation is not about collapsing the distinction between materiality and meaning to advance a specific critical project; it is about historicizing different configurations of materiality and meaning . . . as conditions for the coming into being of a given form of rhetoric" (261). Thus protocols "articulate" network cultures by expressing a rhetoric of relationality, and we can highlight different values across different context. In Chapter 2, I focused on abstraction, and how its historical roots expressed an approach to programming that has been historically gendered and promoted as such. Here, the focus on file sharing introduces a focus on values of individual and collective agency: BitTorrent manipulates and builds upon a performative framework of collective action. But to expand this analysis I also want to incorporate the ways in which identifiable values in these structures articulate in other digital contexts, and to think about how these shared values produce a larger communicative context.

Eve Kosofsky Sedgewick (2002) outlines a notion of the "periperformative." The periperformative is an addition to the concept of the performative, in that it recognizes the performative as an act that calls upon (or is called upon) by norms in each situation, but also recognizes the social "nearness" of the aspects of the performative that can affect (or be affected) by it. An utterance such as "I Do" is not simply a performative in its re-instantiation of the context and norms of legal marriage, but also a call upon those who witness the utterance, who are interpellated by the performative nature of that utterance (70). In the marriage example, the call of "I do" is a subjectivizing act not only for those involved but for those in witness, a call to acquiesce (or not) to the norms of the utterance. She argues that the performative utterance "invokes the presumption, but only the presumption, of a consensus between speaker and witness, and to some extent between all of them [the audience] and the addressee" (69). That is, a performative act calls not only upon the addressed but also those who would witness the act, and in turn spins of multiple potential responses and reactions that not necessarily intended by the original speaker.

The articulation of networks by protocols is not a linear progression of norms from context to context, from technology to technology, but the creation and reiteration of periperformative responses within structured frameworks of agency. Their rhetorical strength lies in their ability then to articulate not only their values, but their efficiency or correctness. BitTorrent, as my main example here, relies on perhaps one of the most powerful discourses in technology: of increased speed and flexibility, of efficiency and correctness. However, the notion of "swarming" is a socially constructed form of participation and collaboration with gendered histories of labor invokes values of

collaborative labor articulate different forms of power tied to business-oriented masculinity, even as it also functions within discourses of social progressiveness.

One example of how a swarm collaborative logic is put into practice is through Amazon's Mechanical Turk. Mechanical Turk is a service provided my Amazon, a sort of employment market for online labor. However, unlike other freelancing sites that exist to link potential employers with contract employees in the areas of technical or professional writing or programming, Mechanical Turk provides services for companies that need "human intelligence" (Amazon Turk FAQ). "Human Intelligence" is nominally a reference to the application of human thought to tasks that cannot be accomplished my machines. For example, typical jobs on Mechanical Turk involve transcription, survey completion, short essay writing, and transcription of words from images. The draw of Turk for employers is that, due to the nature of these tasks being unspecialized and (supposedly) quick to complete, they can offer wages between 5 cents or five dollars, depending on the job. The idea is that a worker, called a "Turker," can complete a Human Intelligence Task (HIT) within seconds or minutes, make money, and do so from home. Turkers initially sign up for an account and search through a list of jobs posted by hiring agencies using the service. Some of these jobs are for student research projects, some of these are from larger companies, some of these are from content providers online. All are looking for an inexpensive way to have individuals complete small, inconsequential tasks. This hearkens to a perhaps-misplaced notion of the modern labor economy as one of freelancing and freedom, where people can work on their own schedule on tasks that they wish to complete. Interviews and studies by individuals show that in reality, working through this micro-labor economy, while advertising itself as a release from money

concerns for those with free time, is a race to the bottom of the barrel in terms of tasks, wages, and time.

Ellen Cushing (2013), interviewing Turkers, found out that it takes roughly 2000 completed tasks to make roughly $150-$200 (Cushing). Employers can reject a Turker's work without explanation, costing the work time, money, and reputation (a staple for the service, as workers with low reps find themselves unable to apply for higher-paying jobs). The primary tasks (up to 40%) of the jobs are also promoting spam and SEO content as quickly as possible, asking workers to quickly visit sites and post responses or write 200-500 word paragraphs heavy in Google-friendly keywords (Cushing).

The problem here is twofold: first, the wages offered for each job are typically not commensurate to the time and effort needed to complete them. Second, wage calculations typically do not include the time it takes to search for jobs and, in some cases, the time needed to gain "Master Qualification" in a given task to find other, higher paying jobs. The freedom to work as one wishes is framed within a network of ultimately disposable labor, as Turkers are paid pennies on the dollar to accomplish tasks that would have normally required actual, hired workers. The swarm model of the jobs, in that each is "inconsequential" or divisible from a greater whole, shapes the ways in which workers are viewed: they also become inconsequential and divisible. How Amazon Turk reflects an engagement with the technology is interesting, because the performative discourse of the distributed network is also the discourse of distributed work. Turk uses swarm modeling to articulate labor precarity across a distributed layer network. As Moshe Z. Marvit (2014) writes, if you happen to be a low-end worker doing the Internet's grunt work, a different vision arises (Marvit). According to critics, Amazon's Mechanical Turk

may have created the most unregulated labor marketplace that has ever existed. Inside the machine, there is an over-abundance of labor, extreme competition among workers, monotonous and repetitive work, exceedingly low pay and a great deal of scamming. In this virtual world, the disparities of power in employment relationships are magnified many times over, and the New Deal may as well have never happened (Marvit).

However, in some cases, the logic of swarms is situated under the friendlier aegis of "crowdsourcing." Aniket Kittur (2010) writes that "Crowdsourcing has worked especially well for certain kinds of tasks, typically, ones that are fast to complete, incur low cognitive load, have low barriers to entry, are objective and verifiable, require little expertise, and can be broken up into independent subtasks" (22). Crowdsourcing, like swarms, is the method of approaching complex tasks as smaller sets of subtasks, with a group of individuals working together to accomplish those subtasks to finish the larger one. Such an approach (specifically under the term "crowdsourcing," and not swarming) are prevalent in areas such as science and independent project funding. In many cases, crowdsourcing has been argued to benefit to projects in which data itself becomes a problem. One such example is HarassMap. HarassMap is a crowd sources locations where women in the Middle East are regularly harassed by men. If they experience some sort of physical or verbal harassment, women can go online and update the map by reporting the harassment. The website uses a map interface, with color-coding that allows users to specify the type of harassment they went through. It also tracks the frequency of harassment in certain areas, while allowing the women to remain as anonymous as possible. This sort of crowdsourcing picks up an endemic problem and addresses the distributed nature of that problem: that harassment occurs repeatedly, and often without

repercussions. Furthermore, this model mobilizes the Internet by allowing women to report this harassment on a map that helps them navigate areas in which harassment may be prevalent. Since a single creator or committee of creators could not manage handling all the incoming reports themselves, nor could they be expected to get information on those reports from police or other authorities (who would probably not share it, share it piecemeal, or never report it to begin with). The crowdsourcing alleviates these issues by distributing the reporting of harassment across the women who are harassed. Participation, in this way, leverages a crowdsourced approach to alleviate responsibility and accountability away from reporting women. The creators of HarassMap describe the project as a way to "engage all of Egyptian society to create an environment that does not tolerate sexual harassment" ("Who We Are").

Through HarassMap, women participate by sharing data about sexual harassment incidents throughout Egypt. Women anonymously submit incidents, which are then coded into a digital map that viewers can see online. Each incident is marked on the map with the time, date, and kind of incident (catcalling, photography, sexual advances). Women who report these incidents are also given a response that indicates places where they can find counseling and legal services in cases of sexual harassment. Since sexual harassment is so wide spread, the map helps to spatialize it: looking at the map, women can see "hot spots" of harassing activities, and relatively safe zones where less or no reports are emerging. Furthermore, it allows women to report such incidents outside of a legal framework and make the knowledge public. As there are often no legal resources for sexual harassment, this service provides another outlet for information to disseminate. This form of crowdsourcing is like the swarm, in that the role of "reporter" is abstracted

from the body of the harassed woman. The agential position of "reporter" is thus enacted in ways that it could not be outside of the logic of crowdsourcing. Since the service is anonymous, women can more freely report such activity. Chelsea Young (2014) writes that "women can share their experiences anonymously, free from the social stigma and accusations of blame . . . the crowdsourcing model also allows women to bypass institutional constraints within formal law enforcement channels that may prevent them from reporting incidents of sexual harassment" (8). This is incredibly effective and powerful, as the onus to complete the task of reporting and identifying trouble areas is placed on the method of sharing itself, rather than the individuals involved, and the designation of "reporter" as an agent position allows women to perform the network: by reporting.

This seems counter-intuitive, but if we think of computer protocols, as they articulate, as performing some sort of expressed communicative power, we can see that even a distributed network of women reporting harassment can be more linear than hierarchical, depending on the vantage point. But the hierarchical definition of responsibility leveraged by HarassMap is what articulates the network of possibility for women to have this kind of agency. And, therefore, we can look at networks like performances of agential possibility, with the norms associated with what constitutes agency already built into the system. Moving back to BitTorrent, we can see a still-linear mode of agency: the users are given a model of participation and follow it. But what is the significance of this modeling?

Conclusion

BitTorrent, as a labor practice, models some basic assumptions about work and participation. Thinking of a network or protocol as a rhetorical agency means framing the abstracting practices within a larger spectrum of activity, one not entirely focused on the rhetorical, autonomous subject of the classroom, of speech, or of writing. The text of the protocol, its specification, is not the primary mode of the protocol in manifesting through networked communication. This is crucial to a rhetorical understanding of a network, because "purpose" here fits into a larger discussion of control and culture. As Galloway writes, algorithms and protocols are first and foremost about control: we do not live in a world of decentralization, we do not live in a world that lacks center or form. Form does not come into play simply through our direct intervention, but as a historical progression of meaning and practice through time. Thus BitTorrent, as a technology that simply "gets the job done" collapses into an understanding of swarm logics writ large across different network models. Amazon Turk and HarassMap are examples of swarms, expressions of a logic that we see operationalized in BitTorrent. BitTorrent, perhaps more concretely and abstractly than either example, articulates the swarm through distributed labor.

This multiplication of individuation produces a swarm of informational bodies performing a given labor. BitTorrent situates itself with nothing but a drive to solve a problem: the problem of sharing files. But extrapolating this logic to other problems, like cheap labor and reporting sexual harassment, illustrates how particular forms of labor, forms that have been historically masculinized in corporate settings, can still exist in seemingly horizontal collaborative structures.

It seems then that what a given goal is plays part in the expressiveness of the network itself, and implicates what might be a rhetorical "audience" and a responsibility

to that audience. As Christian Lundberg and Joshua Gunn (2005) write that "instead of avoiding responsibility, a relentless questioning of the conditions of the production of the agent, and a close textual engagement with the agential narrative and the agent's texts affirms an unconditional commitment to responding, and ultimately to responsibility, despite our corporate attempts to credit the rhetoric community with good conscience" (96). The production of the agent is central to the question of articulation, and stems directly from the expression of gendered labor in programming. A rhetorical question, suggesting an "agent" invite numerous contentious takes on the importance of theorizing (or resisting) an origin to rhetorical agency. But it also invites questions to how agency functions as a product, rather than a source of production, of something like digital communication. BitTorrent articulates various agent positions that express particular and complex forms of agency, forms of collaboration that fit into socially and professionally gendered patterns of power and labor.

Chapter 4:
Trust, Gendered Labor, and Persuasive Network Logics

Introduction

It was only four years after the first major implementation of Bitcoin, in November of 2013, when a single bitcoin traded for a peak value of approximately $1,200USD, a historical high for the new and volatile currency that had fluctuated in value from that peak to a single cent (Watts 2013). As free money activists, libertarian investors, and specialists in privacy and cryptography flocked to the currency, its viability competed with, and often was superseded by, its value as a political and technological commodity. Specified in a technical white paper published by developer and semi-anonymous online user "Satoshi Nakamoto" in 2008, Bitcoin reached its peak exchange value within the context of increasing interest in alternatives to government fiat currency and investment, fueled in part by depression and crushing austerity measures in several European countries. For many, Bitcoin became both a financial enterprise and a political statement supporting the accumulation of massive wealth without government interference. Bitcoin requires no governing institution, which for many provided proved the legitimacy of non-governmental moneys and demonstrated how any problem could be solved or circumvented through the distributed, collective knowledge of the Internet. Bitcoin, one of the first implemented and functional cryptocurrencies in existence, enabled what its creators and users claimed was a free, anonymous, cash alternative for online purchasing. Using a series of conversations of finance, but of privacy circles, European politics

(following heavy austerity measures in EU countries like Greece), and small-government libertarian advocacy (Hern, 2015). Bitcoin's emergence as an increasingly "legitimate" form of money has fueled Bitcoin speculation and technological implementation, but this emergence is predicated on its simultaneous functioning as a digital currency and communications network.

The discourses of government currency, libertarian politics, and political advocacy arise from a larger set of concerns in which Bitcoin, as a technological artifact, functions as critique. To respond to the limitations of online purchasing, the creators of Bitcoin essentially had to create an entire currency, one not predicated on government regulation or banks. But to define this currency, a currency not anchored to an issuing government institution, they created a series of standards known as the Bitcoin protocol that defines what bitcoins are, how they are traded, what kind of security measures protect privacy when they are traded, and how theft and fraud can be prevented while maintaining such privacy and security. The result is a vast network of software, traders, and exchanges that circulate bitcoins anywhere and everywhere in the world. The Bitcoin network, however, is not an amalgamation of users strung together by their shared use of an exchange tool, but neither is it simply a network of rational individuals engaging one another through the economics of technological commerce. The network is, in many ways, its protocol, and this protocol structures the Bitcoin network while engaging in a critical discourse with the structural limitations of networked communication, specifically the protocols that govern Internet connectivity.

As argued in chapter 3, protocological critique as a practice exemplifies discourse that collapses and mobilizes esoteric technology and culture, and Bitcoin is no different. But in building the case for a rhetorical method that describes technological objects like code as having cultural and persuasive mechanisms, I now turn to a discussion of how this mechanism connects implicit ideology with larger, more explicit sociopolitical formations. As Bitcoin is overtly a product of a conservative libertarian politic, it stands to reason that a further investigation of the workings of Bitcoin would illuminate the ways in which that politic permeates and is reiterated through the technology itself. This is particularly apparent when it comes to issues of gender in Bitcoin. This topic comes with some controversy, and not least because the question of gender participation in Bitcoin is divisive, with different individuals and groups providing different perspectives on gender representation in Bitcoin development. In 2014, Popular Science wrote that although the population of Bitcoin users are incredibly diverse in terms of religion, nationality, and background, they were overwhelmingly (95%) male (Popular Science). Citing a study by Lui Smyth, the magazine notes that women in Bitcoin often join for the same reasons (individuality, financial opportunity). Smyth notes, however, that the absence of women in Bitcoin is "essentially a nexus of the tech gender gap, the finance gender gap, and the libertarian gender gap" (Popular Science). Arguments to the contrary often offer anecdotal or friendly reminders that not only are women present in the Bitcoin world, but that *being* a woman in the Bitcoin world is a place where gender is part of a professional identity. Fortune Magazine (2015) notes that there are numerous women involved with the technology, albeit more prominently on the business and investment

side, and they are still in the vast minority of users—and even more so on the production side.

There are a few reasonable arguments for this discrepancy, and I want to focus on one of particular interest: the historically masculinized world of anarchist or libertarian politics and its connection to digital technology mentioned by Smythe. While some argue that social norms funnel men into STEM fields and women into more domestic or care-giving roles, others argue that there are perhaps more fundamental issues affecting how socially constructed gender impacts participation in Bitcoin. As Brett Scott (2014) wrote in a blog post reflecting on his experience at a major Bitcoin conference, some of his friends/fellow Bitcoin users suggest that women are "repelled . . . by the large number of libertarian/anarcho-capitalists in the scene, and the increasingly aggressive culture that surrounds it" (Scott). What he refers to is a strain of libertarian politics that has been a part of software development for decades, tracing back to movements stemming from Free Software development that embrace capitalist models of software melded into an odd mix of open source, crowd-sourced, and corporate structures. Scott writes that the political framework that surrounds Bitcoin often turns to a "libertarian ethos," one that has "the side-effect of attracting those who already feel empowered (or who feel entitled to power)" (Scott). For Scott, this politic is an aggressive display of financial brutalism, a "screw the weak callousness" that fits into a capitalist and patriarchal system.

It is this connection that I intend to trace here. If Bitcoin functions as a protocological critique, then it stands to reason that the connection between the protocol and its politics is in the rhetorical impact of its articulation of gendered, masculinized politics. The irony of the protocol is more fully manifest in its desire for freedom, and the

ways in which its culture (that it generates and is generated from) limits the practice of that freedom in non-institutional ways. I analyze the connections of gendered politics and technologies in Bitcoin to argue that one begets the other, and that the development of political technology such as Bitcoin invites a rhetorical critique of that technology that fundamentally includes gender as a constituent and necessary aspect of that development.

Bitcoin and Logics of Circulation

Before Bitcoin, digital financial transactions could only work through intermediaries such as banks or creditors who guarantee those transactions. Credit and banking institutions verify that funds are available and that buyers and sellers are legitimate before facilitating the transfer of money. Therefore online vendors such as Amazon require credit card or bank account information to process payments, rather than cash or money order. Since banks and creditors mediate financial transactions, they offer safety, security, and reliability that removes the uncertainty of shopping online because they stand in as a mechanism of *trust.* Trust is, in these contexts, institutionalized in the mediating work of these institutions through the structuring of identification, verification, and exchange. In studying Bitcoin, I illustrate how Bitcoin's invention situates it at the intersection of trust in a financial sense and trust in a computer science sense. The former involves how identification, verification, and exchange are all aspects of financial trust, while the latter links financial trust to its translation and implementation within Bitcoin. The connection between these methods of trust will help frame a rhetorical analysis of the Bitcoin protocol.

During conventional online transactions, the identity of a purchaser is confirmed via credit card or banking information, funding is verified via the financial institution, and exchange occurs at the behest of the purchaser through the institution. Along with this, several further checks come into play: purchases can be traced (by both by date and location) and, in cases of fraud, contested. Trust, in this sense, is vested within the institutional capacity to keep records of (and verify) buyer and seller identities, to verify the existence of money or credit to complete the transaction, and to implement the exchange of this money between the buyer and seller. There are a number of ways that banks do this, usually through credit card or account numbers linked to verification numbers and banking identities (typically tied to a user through a social security number and physical address). As cybercrime and threats of international terrorism have increased, banks have strengthened their security measures to include secondary PIN numbers and security questions. Subsequently, online marketplaces have implemented point-to-point encryption standards as part of their online purchasing platforms. The communication of account verification information is heavily protected both at the point of purchase (during the act of transmission) or for further purchases (typically when a customer is a regular user and shares banking information with the company for repeated use).

Bitcoin, however, replaces the required trust of third-party institutions with a system of mathematical checks and public regulation, and does so through its implementation of distributed control and security protocols. To begin, a bitcoin is not a "coin" in the sense that there is a minted physical object of exchange. "Bitcoin," instead, is an abstraction of a chain of authorized, encrypted transactions between different parties

that serve as a record of that coin's history, which verifies that the coin belongs to a specific person at a specific time (Nakamoto 2008, 3). So, if I have a coin, what I really have is the legitimate record of all transactions leading up to me that verify that I indeed own the coin, with my receipt of the coin ending that chain (until I trade it). I hold and exchange that bitcoin through a piece of software called a "wallet," which identifies me as a node in the Bitcoin network, an alphanumeric string of characters called an "address." Each address is unique to a wallet, not a computer, and requires no personal information to use.[3] However, the address *is* used in the identification process. During each transaction, the Bitcoin wallet software encrypts data from the transaction through a method known as Public Key Encryption. What PKE does is utilize some esoteric aspects of prime numbers to create pairs of related keys for each user: a public and private key. When I send bitcoins to another user, I encrypt this data using my private key that no one else has access to. The only way that the user can decrypt this information (and get their bitcoins) is to use my public key, which is available to everyone. If the user decrypts information sent by me with my public key, this means that (unless my private key is compromised) the information was sent by me. Furthermore, each transaction also includes, via PKE, a digital signature tied to information within the transaction that can only be verified with my public key (2).

[3] As a comparison, imagine a merger between a physical wallet and a bank account: you can take the wallet with you (on a USB stick or something else) and use it anywhere, but it has enough information to connect to the financial network to send and receive money. So long as the owner of the wallet does not overtly or implicitly identify himself or herself, the only public record of the user is the wallet address. This means that I could use my wallet to trade bitcoins at a home computer (which would make them traceable to me personally), or on an encrypted network, VPN, or some random Internet cafe (which would make it more difficult to trace to me).

The wallet, upon installation on a computer or external storage device, syncs with the public Bitcoin ledger, which verifies all the coins currently in the network. The ledger, called the "blockchain," contains all the dated and verified transactions that have occurred over the lifetime of the Bitcoin network, and identifies which bitcoins belong to what wallet address. The verification of funds for each user is therefore vested within the blockchain, which exists as a distributed record of information stored by every node/wallet in the network (2). But the maintenance of the blockchain, rather than the responsibility of an institution, is distributed through the network by way of bitcoin "mining." A miner is a user on the network, but one that donates CPU time from their computer to help solve the complex equations Bitcoin uses to protect transactions. Since Bitcoin exchange happens in real time, what miners do is, through their own computers, solve what the Bitcoin protocol calls the "proof of work" concept to verify transactions and update the blockchain. Avoiding the complexities of this proof of work concept, what ends up happening involves a race between miners to verify transactions: once a miner completes the proof of work necessary to verify a transaction, and if that transaction works with the current blockchain (that is, that the blockchain has not already been updated with the same or conflicting information), the transaction is placed in the blockchain ledger and the transaction is now verified (4). To incentivize the verification of transactions that allows them to be added to the blockchain, the network (of its own volition, as part of its algorithms) releases a set number of bitcoins to those who donate their CPU processing power towards verifying transactions (4). With encrypted identification as a standard, and the public blockchain as a verification schema, the

protocol, through wallet exchanges, allows the transfer of bitcoins from any wallet to any wallet, so long as they are connected to the network.

Trust, rather than being placed in an institution, is operationalized in the protocol. Therefore, the Bitcoin protocol must find a way to articulate trust in the financial, political, and technological contexts that best express its ultimate goals. The benefits of identification, verification, and exchange find their antecedents in the Bitcoin protocol through the operations that it dictates, the standards that it implements, and the transactions that it then facilitates. Bitcoin users do not have to worry about fraud and double-spending because all transactions are public, and every wallet has an updated version of the ledger. Miners maintain the integrity of the system by performing complex encryption verification without a central agency to mediate that verification. Trading and mining are also decentralized: with the correct software (or hardware, in the case of mining), anyone can trade or mine, and any node can push an updated blockchain ledger so long as it adheres to the standards of the protocol. Since the points at which bitcoins can be traded are diverse, as are the hardware and software configurations used to mine bitcoins, and (in the case of mining) are often distributed across cloud-computing platforms, the ways in which Bitcoin allows for the movement of bitcoins extend across various registers of cultural and technological circulation.[4] Since the release of bitcoins and the verification of transactions gathers the generative, shared agency of the nodes on the network for its own maintenance, the dispersal of ways to mine or trade bitcoins also

[4] While independent mining hardware exists, and is deployed by hobbyists, the increasingly competitive field of bitcoin mining has led to the rising market of cloud mining, where a group of miners will work together, often purchasing or just building distributed cloud processing, to pool computational resources and, in theory, have an edge over smaller groups or individuals with less computing power.

harnesses this distributed agency to perpetuate the network. As each node has “no chain of command,” there are “only autonomous agents who operated according to certain pre-agreed 'scientific' rules of the system,” and as such each node shares the responsibility of maintaining the Bitcoin network (Galloway 38). The mechanisms that perpetuate this network are ultimately repeatable by any node in the network, so the circulation of bitcoins as a currency reflects the repeated, distributive movement of verification, mining, and trade.

Since the Bitcoin protocol dictates the distribution of network maintenance across all its users (or in network parlance, “nodes”), it also distributes the responsibility of the network’s continued existence to those nodes, and as such defines the structure of those nodes (how they are embodied on the network, what sort of practices they engage in). Distribution as a network mechanism resembles other protocols, and illustrates how Bitcoin resembles other communication networks, and how protocols function as mechanism of control.

Part of how Bitcoin articulates trust is how it incorporates the limits of TCP/IP. TCP/IP recognizably engages in a set of distributive practices similar to those of Bitcoin—the distribution of necessary actions to each participating node in the network for upkeep and network maintenance. TCP/IP’s existence (and, consequently, the existence of the Internet) demonstrates the strength of distribution, because its logic of distribution keeps the network robust and open. However, TCP/IP’s continued self-maintenance also works through another piece of logic, in that IP transmissions are non-anonymous. The protocol dictates that IP packets contain data recording the sending and

receiving addresses so that packets get where they are going. All data packets pass through multiple routing computers to get to their destinations, and intercepting or rerouting packets of data based on their destination and origin has become increasingly common (Gavrichenkov 2015, 3) This is a huge problem for online cash, since the manipulation of data could result in the loss of thousands of dollars through direct theft of bitcoins. But even more importantly, Bitcoin, in articulating trust, also sought to mechanize privacy: trust did not necessarily mean a personal accountability to anyone but *did* mean that the logics of trust in place could exclude the need for unwanted exposure over insecure communication lines. So the Bitcoin protocol defines a network that plays loose with IP addresses by separating the necessary connection between an IP address and the user of a wallet. Bitcoin users can connect their wallet to the Internet at any location. While the wallet and its use will be broadcast from that IP address, the connection between the user's identity, the address, and the wallet address remain separate. Alongside technologies such as a Virtual Private Network, Bitcoin offers a feasible (if not necessarily complete) privacy solution. With the proper encryption in place, not only are transactions private, but the identifying link between a node's location on the Internet and a wallet address is broken. Furthermore, Bitcoin leverages the non-anonymous aspects of TCP/IP through the blockchain. Since all transactions are public, all nodes are visible and all expenditures are traceable. But without a direct link between the node and its location, this extreme publicity becomes irrelevant: its institutes trust and privacy in one fell swoop. Thus, not only is Bitcoin like a cash for digital purchasing, it maintains the privacy of cash and the trust system of a governmental currency.

A common term that gets used quite often in this conversation is "distribution," one that I believe centralized the political, communicative, and ultimately gendered question of agency and rhetoric in Bitcoin. Distribution not only serves as a productive fulcrum for thinking about how rhetoric and writing work in complex rhetorical situations, it also introduces an important facet of protocological order—control—another important concept for understanding how digital networks of circulation operate. We can think of circulation as a distributed affair of rhetorical production across a variety of networked sites. Distribution, in rhetoric, has typically referred to the way a text is prepared or delivered and circulated in and for a network of readers or audiences. The definition of distribution has changed, however, to address the complexity of writing publics and technologies, and often refers to the distribution of a writing process or rhetorical encounter across a variety of times, locations, and media. Jenny Rice (2007) writes that the shift from static models of rhetoric to complex ecologies opens up our attunement to writing as a "vocabulary of distribution'" that speaks to how writing practices are distributed across a material-social field of action (13). Rice's focus on distribution points to how the "site" of rhetoric is less a discrete place and more a series of dynamic events and shifting, fluid locales of rhetorical interaction and production (10). Such a notion of distribution seems to share some affinities with James Porter's (2009) attempt to inscribe distribution within a framework of digital delivery. Porter describes his idea of distribution as the production of texts and the choices of authors geared towards the idea of affecting the circulation of those texts. For Porter, distribution is closely related to circulation in digital technology because distribution, defined as "rhetorical decisions made about the mode of presenting discourse in online situations,"

implies thinking rhetorically about future circulatory possibilities at the point of production (214). For Porter, circulation is also the movement of rhetoric, or the circulatory force, that can be understood as a continual process of redistribution. Circulation, he argues, refers to the "potential for [a] message to have a document life of its own and be re-distributed without your direct intervention" (214). The document life of a message (in this example), he wants us to understand, is made possible by its redistribution across different rhetorical agencies, which implicates those agencies and their ability to redistribute that message.

Because circulation implicates a distribution of action across different points in space and time and across different acting agencies, it invites us to think about a model of distributed agency. Borrowing from Bruno Latour, many rhetoricians have used his concept of distributed agency as a key aspect of their theoretical work. For Latour (1999), there are never individual acts, only the distribution of action across different acting agents. There is also, therefore, no single locus of agency, nor a pre-defined definition of agency, but only networks of distributed agency and transformation (306). Distribution opens up, then, not only the question of how, and by what means, rhetoric circulates (which is in large part the reason many theorists are inspired by the idea in the first place), but also introduces the method of "tracing" the relationships of circulating actors—what Jeff Rice (2008) describes, during his Google Maps-inspired travels through Detroit, as the "figuring out of agency and relation in a given situation" (28).

Laurie Gries (2013), for example, in following the *Obama Hope* image produced during the 2008 presidential campaign, developed a method called "iconographic tracking," which is indebted to Latour's call to "reassemble the social" through the study

of distributed rhetorical agency and circulation (2005, 333). Suggesting a network approach to the study of how the *Obama Hope* images circulates, she writes that

> once produced and distributed in a networked pathway, images rapidly undergo change in terms of location, form, media, genre, and function. In addition, as rhetoric emerges from an image's encounters with human and other entities, images are often catapulted back into flow in divergent directions and generate even more configurations. (335)

The difficulty, argues Gries, of studying something like an image as it circulates is that we are rhetorically trained to view it *as* a discrete object, a remnant of a heritage of representational assumptions of the stability of meaning. Gries would rather have us recognize that tracing an image is less the tracking of a moving object and more the tracing of transformations and rhetorical processes across redistribution points.

The communicative labor of a network like Bitcoin lies directly in the distribution of communication roles, and not the other way around. It is the logic of the protocol that defines and articulates those boundaries and activities, then framing a technological context for the circulation of value. Bitcoin's value is represented in bitcoin nominally, but more expansively the value of trust of circulated as a political phenomenon: it is not just a currency, but a form of freedom born from a distributed form of control. The protocol articulates bitcoins, articulates miners, but also articulates a political efficacy grounded in individual pursuit of wealth, constrained participation, and the value of our faith in the infallibility of digital technology and mathematics.

The Politics and Rhetoric of Bitcoin

The politics of Bitcoin then will focus on a relationship between the logics of the politics at play in the development and use of Bitcoin. These are co-constitutive, and reside at a fault line between rhetoric as a political action and rhetoric as a technological analytic. Bitcoin's politics, much like BitTorrent's, are an ironic reconfiguration of trust that mobilizes certain values through the exchange of mediating institution for a mediating structure and algorithm. In the first, institutions, ostensibly filled by clerks, bankers, etc., are an essential part of interaction. In the latter, the middle 'man' is literally removed, and the burden of legitimacy placed on a seemingly normalized and transparent method of exchange. This is not to say that the human has been replaced by the machine: large banks as a matter of course incorporate complex software that ensures safe transactions and raises warnings when illegal spending or identity theft seems to have occurred. What it does say is that the politics of trust are extensible through different technologies. Trust as a value is not an intact entity that persists over time and space, but it is instead a value of the system, articulated by the system, and finally finding itself as a performative aspect of a larger network of users that it structures.

What does it mean to say that Bitcoin articulates a politics of trust? Answering this question involves tracing numerous aspects of the technology and its effects. The most immediate is the politics of privacy and individuality that are present in the Bitcoin white paper published by Nakamoto. First and foremost, Bitcoin is a critique of federal banking systems and regulation. This critique is leveraged through a digital medium precisely because that is the platform where trust has been in its most intractable state. Online transactions *necessitate* regulation and the middle-man logic of trust because of the inherent anonymity of the Internet. I cannot trust cash-like currency in digital

platforms because there is no way to completely ascertain who is trading, and there is no way to verify the exchange of a physical commodity through a digital medium without some sort of mediation. In Chapter 3 I spoke of the TCP/IP protocol. One of the interesting facts about TCP/IP is that the required and distributed routing action that occurs as part of network formation also requires the general knowledge of IP addresses across the network. That is, when I pull data from a server I must know its address, and it by default knows mine. But every computer that aids in the routing of packets across the network also knows them as well, and as such if my IP address requests data from another server, there is a routing record of my activity. TCP/IP is inherently non-anonymous, and while there are Virtual Private Network (VPN) and encryption packages that attempt to counteract this, the truth of the matter is that they are incomplete solutions built on top of the already-existing limitations of the protocol.

Anything built on or using TCP/IP must contend with this issue, especially if it offers any sort of security. To do so, Bitcoin disconnects a single aspect of TCP/IP (the IP address) from a necessary aspect of Bitcoin (the Bitcoin wallet address) by making the ledger public. If there is no connection between my wallet address and any given IP address that I might use (at an Internet café, through the TOR network), then the potential for secure action increases. But security is inherently limited, and so the important articulation of the technology is not security but trust in the security system. The trust comes not from the verification of my identity, but the validation of the exchange through the swarm of miners in the network. This "distribution" of the responsibility of network functionality on the surface mimics the distribution of routing responsibility in TCP/IP. However, Bitcoin subverts TCP/IP by leveraging swarms to solve its problem of trust.

There is no longer a need for institutional support when the entire network is the institution, but the institution of Bitcoin distributes trust as a rote operation of brute-force problem solving. The institutional role of Bitcoin the protocol melts into the background of a distributed network as the value of trust is grounded in a mediation and redefinition of agency and a shifting definition of communicative force as a legitimate exchange of value (value which is also constructed and circulated through the network via repetitive and self-constructing distributed action). The circulations of bitcoins are quanta not of economic value, but of exchanges of trust and communications of share agency.

Bitcoin not only critiques regulated banking, but mobilizes the critical BitTorrent model to address limitations of the underlying technological structure at the same time. Bitcoin uses trust to then critique the model of trust in TCP/IP, transform is, and mobilize it in a different political technology. Ronald Walter Greene (2004) writes that so long as "rhetorical agency as communication is harnessed to a communicative model of interaction, rhetorical studies are destined to lodge a permanent anxiety over the quantity and quality of rhetorical agency needed to change the structures of power at the center of its intellectual labor" (203). The constant "rescaling and reframing" needed to sustain this engagement between individuals and political institutions leads to rhetoricians becoming "moral entrepreneurs" for right action, defining effective moral behaviors for communicative action (203). Greene and Nelson (2014) argue that digital rhetoric serves as a reinvention of the concept of communicative labor, in that it is not only the strategies and models of communication that produce value, but the labor of communication itself and the harnessing of value therein.

Gender Politics and Queer Performance

The distribution of a political agency seems to place an intense locus of control on agents within the system, one that determines actions and actionable space (or lack thereof) in the use of the technology. Distribution is the engine of circulation, quite literally in this case—the distribution of nodes defined by Bitcoin enables the circulation of bitcoins. But even beyond this, the definition of distribution *as an action* illustrates how circulation functions as a rhetorical practice, and furthermore how the technologies of circulation are more than mere tools (and, perhaps, more than extensions of people using them). But beyond this, I argue that the control begets subversion as two necessary sides of the same coin (pun intended). The mere existence of Bitcoin as a multimodal critique of technology and privacy as well as a financial banking system attests to this, and through the performative chain of technological structure, with one technology emerging and reifying its predecessor and articulating its future potentiality, technopolitics emerge.

In Chapter 2, I quoted Zach Blas who described his work as a "disidentification" from capitalist technology. This term was created and mobilized by Jose Esteban Munoz (1994) as a description of subversive performance. Disidentification is not a rejection of normative identification, nor an acceptance, but a "rewiring" of the logics that maintain those identification as necessary or normal is society (9). For Munoz this was all about performance; citing performance art and artists working in queer and drag cultures, he highlighted how the performance of gender and sexuality was at the heart of disidentifying with dominant heteronormative culture. More concretely, it is the process of the "decoding of any mass, high, or other cultural field from the perspective of a minority subject who is disempowered in such a representational hierarchy" (26). The

disidentification that Blas mobilizes is the ideation of a sexual or gendered minority *of* technology, or specifically within the labor required to build it. The technique of abstraction was not only an effective programming technique, but an historically gendered one that marginalized women almost completely out of the field of programming. But disidentification also provides the platform to talk about what agency looks like within these protocological spaces precisely because in "rewiring" the logics of a dominant culture, the agent must practice some of the logics in new ways, and therefore must also indebted him or herself to those logics in a way that promotes their use. That is, the potential for disidentification, and rhetorical or political action, is rooted in adopting and subverting distributive logic.

In the case of Bitcoin the performance of the minority seemingly begins with the adoption of a seemingly apolitical and gender-neutral majoritarian position. Brett Scott argues that one of the biggest myths tied to the Bitcoin development community is that the technology itself is simply a platform in which all autonomous and self-sufficient individuals are allowed to participate—but that participation requires effort, labor, and commitment (Scott). David Golumbia (2015) argues that Bitcoin is itself buoyed on a sea of "cryptopolitical" thinking that supports ideologies of self-determinism, radical isolationism, and a vague and abstract notion of "freedom" that endears it to a right-wing libertarian politic (123). The issue with these politics is that they, in their manifestation in technologies like Bitcoin, emanate from an historically white, male class of technological producers that emerged from the convergence of digital programming and academic achievement post war. What is "normal" in these communities is this whiteness and a kind of masculinity tied to libertarian politics tied to notions of meritocracy and

achievement. Golumbia notes that it is the volatility of the currency, tied directly to its libertarian underpinnings, which limits it to a very privileged and wealthy class of investors (124). The tool itself is a product and production of privilege, and more specifically privilege associated with a white heteronormative business culture that values self-determination and freedom. But self-determination is a two-edged sword, and is often the mode in which oppression is practiced. In a point-counterpoint editorial framed around the question of "is libertarianism a boon for LGBTQ individuals?" Julie Novak and Simon Copeland (2014) discuss the personal and structural aspects of LGBTQ oppression in society, and the compatibility of that oppression with classical libertarian ideals. Copeland specifically points out that the notion of "individual" freedom does not necessarily equate to personal freedom to act, writing that "much of this structural oppression is imposed by the state" (Novak and Copeland). Furthermore, the notion of a free market as one that liberates individual agents to act is at odds with the notion of individual liberties, particularly in instances where institutions or business make decisions that disenfranchise individuals in that market. Bitcoin is at its core an algorithmic representation of a sort of freedom of exchange, free from regulation outside of a group of peers. At the same time, it is also a marginalization of that freedom to act through the work of that peer group. Furthermore, the definitional nature of protocol in defining the nature of that peer group further restrains and controls the potentials for marginalization in that group.

Freedom to act in this sense does not mean the guaranteed freedom of agency. Erin Rand (2014) argues that agency, and rhetorical agency, is best framed in a way that understands "agency as emerging from a set of determining forces external to the self,"

and that "it is the formal features of text . . . that enable rhetorical agency" (20). Rejecting the notion of a deterministic understanding of agency, Rand argues that agency and invention emerge from social relationships of power and the formal elements of texts that are bound up within those power relationships. This is where political action—and in her argument, queer political action specifically—both recognizes and pushes against traditional notions of agency that rely on the moralizing idealism that Greene critiqued. Therefore, Rand argues that "rhetorical agency resides neither in the rhetor nor in the text but arises from the positioning of discourse in terms of its formal features" (21). This is important for political activism, and LGBTQ activism, in that the field of political mobilization and resistance is not always tied to the moralistic entrepreneurship Greene critiqued, but instead finds outlet and recourse through the modality of expression and the forms that utterances can take. To say that Bitcoin is a critique is to say that, on the one hand, it is an enactment of this kind of understanding of agency, a realization of political possibility (or lack thereof) in form and structure.

On the other, it is to recognize the critical work of queer and feminist rhetoric in disentangling digital media from the discourse of neutral tool-use. Jacquie Rhodes and Jonathan Alexander write that "queer rhetoric names a constellation of discursive practices that emerge at different times for different groups in order to articulate resistance to regimes of sexualized normalization," which in part calls into question the focus on gender and practice in my work. However, it is precisely the normalization of heteronormative practice as a technological practice that then articulates gender disparity in technological work. This is because protocol is a rhetorical form, as demonstrated, that has a single purpose of defining modalities of agency. The gendered politics of Bitcoin

thus come into relief through the question of agency, and its grounding in libertarian politics. Novak argues that the strength of libertarian politics is the freedom to break the mold of subjectivity in society and thus present a new form of resistance. However, as numerous queer theorists have pointed out, "breaking the mold" usually limits political action to a single actor and his or her relationship to an oppressive force within a direct situation. Copeland writes that "the growth of the modern nuclear family . . . was formed under capitalism, and is certainly queerphobic" (Novak and Copeland). By pointing to marriage as a structural limitation of agency for LGBTQ subjects, Copeland corresponds with Rand's assertion that the form of relationships is such that agency is directly informed by, shaped by, and controlled by the formal discursive structures in which we relate. Marriage, as an example, can play out as a movement for civil rights, but it can also establish and reiterate conservative notions of familial units. Likewise, a political structure like Bitcoin that seemingly opens up a plane of action outside of regulation simply succumbs to a different sort of regulation, not of governments but of majorities. The gendered politics of Bitcoin are such then that they describe an imaginary space of action free from gendered politics, even as the ideological groundwork of the protocol and the politics (libertarianism) faces increasing inquiry into its lack of diversity.

The protocol of a given network system is itself a definition of agency, and it articulates the boundaries within which effective action can occur. Rand writes that her definition of queer "works against the prevailing academic and popular trends to employ 'queer' . . . as an umbrella term for 'gay, lesbian, bisexual, and transgender' identities . . . This is a de-essentialized notion of queerness that disconnects 'queer' from any particular referent and refigures it as the undecidability from which rhetorical agency is articulated"

(22). She also argues that this move still grounds political activism with a focus on sexuality and politics without being beholden to representations of sexuality. In terms of computational activism, this allows for a discussion of computational artifacts like protocols as both political and, if not explicitly sexualized, then implicitly gendered. But it also allows for a discussion of them as rhetorical objects that are not situated in political modes of rhetoric critiqued by Greene. That is, rhetoric is not, as Cheryl Geisler (2004) argues, bound to civic and educational models of persuasion to maintain relevancy, but instead an elaboration of agency as it is articulated by rhetorical media.

Conclusion

This leaves us with Bitcoin, not as an abstract protocol or system of software, but as a rhetorical framework that includes (but expands beyond) the context of textual interpretation and invites questions of rhetorical and political action. Its greatest political statement, very much a parallel to languages of efficiency and correctness in computational cultures, is its ironic assurance of its non-political enactment. On the surface this seems illogical: how does a technology with an explicit political agenda function as a non-political technology? Because the rhetorics and agencies that emerge from a persuasive object of discourse like the Bitcoin protocol are not divorced from their historical and cultural context. Bitcoin is a conservative political project that carries with it a baggage that it articulates through its use and its users. It functions within a political context, but it also constructs that political context by legitimizing certain forms of labor and communication. The lack of women's perspectives in the community can be traced to the community that surrounds it. Such a tracing brackets the technology outside of such

conversations and, ironically, allows for the apolitical myth of the technology to proliferate even as it articulates political agents. The drawing of boundaries around Bitcoin as a technology is part of its articulation of itself. It isn't simply a question of acknowledging that technology is cultural, but better understanding how technology *functions* culturally.

A key question here is "what does it mean to call Bitcoin a conservative technology?" We can begin to unmask some of the normative assumptions of the technology first and foremost in our understanding of it as a tool. Language and rhetoric can and are framed as tools of expression quite effectively in specific contexts, but when attempting to map this into all contexts we lose sight of what it means to define rhetoric as a relational practice. When we think of a tool we think of externality, of intention-driven work in which the ideological or cultural aspects come from a joint application of intention and tool, of will and means. But with Bitcoin, or perhaps most protocols, the tool is itself a vast cultural network of ideological assumptions. Bitcoin in particular highlights what Galloway argues is one of the central aspects of protocols: that they function best when they hide their own inner workings. Abstraction, articulation, are all methods of hiding that require a sort of hiding of the underlying mechanisms that enable and constrain communication.

The question of gender and resistance here necessarily extends beyond representation, because critiquing representation in a field such as cryptocurrency and Bitcoin allows for the myth of apolitical labor and technology to persist. Connell and Messerschmidt (2005) describe how, in certain forms of social relations and workplaces, certain forms of masculinity circulate and fall into stabilizing patters that they call

"hegemonic masculinity" (831). This is not a concrete definition of the masculine subject, but instead an analytic of how masculinity is constructed and solidified through particular social practices, while still allowing for different kinds of gender and sexual expression. Hegemonic masculinity is clear in a community like the Bitcoin community, because while there are generally alternative forms of masculinity on display, they fall into stable patterns that rest on notions of self-determination, resistance to social accountability, and a lack of concern for (or outright hostility towards) not equality, but equitable conditions that would inform equality in a productive fashion. It inherits this from BitTorrent because BitTorrent is a technology that articulates an exchange of responsibility and accountability, a transformation of values from one space to another. Thus when Winter Trabex writes a critique against a Facebook post discussing a "trans-inclusive bank" (or a bank that outwardly and rhetorically includes a group marginalized by sexuality and gender identity), she does so through a lens of neutral capitalism. So when she writes that "bitcoin is the [most] inclusive, pro-equality currency that has ever existed", they do so unapologetically through the lens of neutral currency (Trabex). This is because Bitcoin "does not care if you are gay/Bitcoin does not care what country you are from/Bitcoin does not care if you have breasts and a penis, or if you are a man with a vagina" (Trabex). Truly, Bitcoin "does not care" if you have a penis or a vagina. Bitcoin may not advocate for a specific sexual norm or gender identity. But Bitcoin's expression is far from apolitical, and its politics extend out in ways that go beyond strict representation. Bitcoin is not just a technology for conservative or capitalist use; it is itself a conservative and capitalist technology.

Chapter 5:
Conclusion

The imperative to think of digital technology as a form of rhetoric stems from the necessity to think of such media as inherently political and, in my analysis, gendered. Such an imperative shifts slightly from different theories that suggest that technology shapes already-existing discourse, or conforms to particular models stemming from spoken or written communication. In many ways, technology does just that. But it only does so when such theories view digital media through the lens of directed, persuasive language. If viewed as an expressive artifact that connects different forms of labor and collaboration, different forms of consumption, and multiple kinds of political performance, it manifests as a form of discourse in and of itself. And with such a manifestation the gendered (and, in many ways, the more generally ideological) implications become much more apparent. The claim that technology is political and gendered is therefore supported by such an analysis because it moves the burden of proof away from a strictly textual, or even strictly procedural analysis to incorporate a broader analysis that investigates technical documents like code and protocols as inherently rhetorical components of larger communication networks. I would also contend here that scholars looking to effectively analyze and utilize this kind of discourse must better understand the nuances of the given technologies at hand. Such an understanding, I argue, comes through some changes in scholarship and professional practices that I will discuss here.

In my argument, I have argued for the analysis of the expressive potential of computer code and protocols through their capacity as relational technologies. "Relational technologies" are technologies that are first and foremost technologies that structure communicative relationships through their logic of operation. This at first glance seems self-evident. As Marshall McLuhan (1966) argued, "[a]ll media exist to invest our lives with artificial perceptions and arbitrary values," suggesting that the foundational act of a medium is to produce "values" (199). While there are some implications in McLuhan's work that significantly differ from some of my findings here, I do support the idea that technology creates and propagates values. Relational technologies construct relational values, certainly, but they do so because they circulate at a level of relationships. Styles of labor and communication beget logics of the same, which structure global networks, all of which perform and reiterate the value inherent in those styles. If a style of collaboration tends to favor characteristics that have been associated with particular genders, then we must recognize those styles as part of the discourse of code and protocol even if "gender" as a textual or linguistic construct is not present in a given technical document. Code and protocols are designed specifically to structure the potential for further communication, either through the shaping of coding language and syntax or through the shaping of labor and agency in communication networks.

I focus on constructions of gender because they represent a major, though not exclusive, problem for technical communities. Gender highlights the horizon through which technical specialists can adhere to calls for correctness of efficiency—that is, the inherently apolitical in such communities. The notion that politics are a performative aspect of computer code and networks links gender performance and computational

techniques like abstraction as historical rather than (or not simply) mechanical. Grounding that claim in the labor and vocabulary of software engineering and computer science further supports claims by scholars of feminist technoscience (Wajcman 1991; Barad 2007; 2007), cyberfeminism (Haraway 1991; Daniels 2009; Wajcman 2004), and queer computation studies (Blas 2008; Gaboury 2013) who argue that technology is inherently gendered and political. In my analysis, I illustrate how abstraction and erasure in programming reflects techniques that are apparent in reading technical documents like code and protocol. I suggest that these techniques cannot be separated from the labor and collaboration that produce and utilize them, which in turn situates programming and technical documents in a larger history of performative gender. Likewise, queer activism in digital technology cited in my argument further demonstrates that this gendered and political aspect of technology reconnects the seemingly disconnected aspects of software development, namely contemporary programming culture and the products of that culture (such as software, protocol specification, etc.). Queer activism in digital media highlights the ways in which digital technology, gender, and even sexuality overlap as the politics of a given community. My argument expands on this to show how a particular form of masculinity is historically constructed within a technological continuum of use and production. The practices that structure such technology are rooted in very simple assumptions about how gender conforms with different kinds of work, which then impacts programming communities, global businesses, technical education.

I believe that understanding the rhetorical capacity of technology in these terms presents great opportunities and challenges for scholars in multiple fields like rhetoric,

gender studies, and media studies. I generally break these challenges and opportunities into three general and interrelated categories: theory, teaching, and collaboration.

Theory and Analysis

If digital technology is political and gendered, then successful analysis will grapple with the terms of that technology in order to articulate how that works. In the "terms" of computer code and protocols, scholars cannot relegate a gendered practice only to textual or procedural representation without embedding that text or procedure within its historical and technical context. The myth of correctness or effectiveness in code rests in the fact that it is meant for processing by a machine, which implies that there is a knowable limit to how code can function culturally. The shift from one form of communication to the next will allow us to recognize how, say, rhetoric or gender studies informs analysis of technology. But conversely, the medium of expression dictates new terminologies outside these theories that will inform how we think about things like persuasion or gender. For example, discussing code through rhetorical terms offers insight into the persuasive nature of the language of code and of algorithms while, in many ways, reasserting objects like code as singular texts with recognizable authors, audiences, and intentions. As Katherine Hayles (2005) argued, code cannot function as performative language in the terms outlined by thinkers such as Jacques Derrida or J.L. Austin because of the limiting context of the machine (55). I counter that positing the machine as the premier limitation of code *qua* discourse (pun intended) is self-fulfilling because it maps the limitations of language onto code without augmenting its assumptions for that medium. Rhetorical or discursive approaches to code that already

presuppose the machine as an "audience" or arbiter conflate correctness with affect specifically because the mechanisms of computation are ultimately seen as outside cultural interpretation, rather than co-producers of it. Feminist and queer studies help reframe the question as one of labor and collaboration rather than of discourse so that a given piece of code can at least be acknowledged as only a small part of a potentially monstrous piece of software, which itself is embedded in a long history of code and programming.

Theorizing code as gendered and open to rhetorical analysis will ultimately mean focusing on programming as writing, which in turn means actually grounding analysis in those practices. Programming is in fact a form of communicative labor, one with connections to traditional rhetorical frameworks. But it also represents a unique context of communication with different norms, goals, and participants. An algorithm is not an algorithm without the bedrock of knowledge, practices, and usages that give it its context. Rhetorical approaches to code highlight the impact of software on rhetoric while defining the scope of code as a form of communication. But by approaching code itself as performative, we can address how limiting this can be, particularly when attempting to specify how something like gender becomes part of technology rather than something alongside it. This is why the initial move of recognizing performativity (as outlined by Derrida), gender performativity, and computational abstraction as incredibly similar phenomenon in very different contexts is so important. Analysis and theorization will see procedure and text as markers that perform particular forms of labor, rather than strictly representational texts, which incorporates a larger and wider set of concerns for users, engineers, and even the machine.

Collaboration

To do this kind of analytic work, scholars in rhetoric and gender studies will necessarily begin to collaborate more closely with scholars in fields like computer science. I imagine this move as analogous with the shift from lone programmers writing small scripts and utilities to large, industry-supporting or defining software. The scope of collaboration and labor needed between these two extreme ends reflects differences of kind rather than scale. As Fred Brooks (1975) acknowledged in the very earliest days of software development, "[w]hen a task cannot be partitioned because of sequential constraints, the application of more effort has no effect on schedule" because the application of more personnel hours is essentially counterproductive for non-sequential labor like software development (16, 20). Moving from one end (smaller development) to the other (larger development) is not simply a multiplication of responsibilities and workforce, but a fundamental shift in how people work together on a single project. In investigating algorithms or small pieces of code, I fear that we fall into the trap of mapping our own expectations on a process that doesn't bear them out at larger scales. However, I also reject that as we move into larger scales of complexity that theory drops focus on the technology altogether, or roots analysis only in the social uptake of a technology.

Thinking about larger-scale software production calls for an incorporation of the techniques and labor that emerge from such fields, which requires a much closer working relationship with experts in those fields. The above productivity-scale problem serves as a metaphor for the gap between how scholarship works in different disciplines. While humanities scholars have made great strides within the contemporary model of scholarship that favors individual or small-collaboration publication, it may come to pass

that more expansive investigations into code and other technical specifications will not remain feasible outside equally expansive approaches to collaboration and publication. This is the trap that I see myself fall into even as I write this: my approach as an outsider—as a humanities scholar considering the realm of digital technology through particular understandings of the political work of technology. I may have familiarity with the field, its terms, and its concepts, but it has been years since my undergraduate education in computer science, and years since I have done any serious coding. And, having never worked in the field, I have come to understand that this is much more of a limitation for my work than when I started. To condemn a toxic, misogynistic worldview is necessary, but the next necessary step is to engage through shared values and new, interdisciplinary approaches to scholarship. If styles of collaboration reflect historical constructions of gender in the technical fields that I have discussed, then we can also model and perform different styles of scholarly collaboration that reflect different sets of values, different political assumptions. One might argue that these steps have already been taken in areas like digital humanities. As Adam Kirsch writes, however, there has been focus on the building of tools and methods as an expansion of the field that simultaneously marginalizes other portions of that same field (Kirsch). In omitting the focus on building tools or serving as clients for computer scientists to build software, I believe that critical scholars of rhetoric and gender can address the essential questions gender, technology, and labor raised by my argument.

I believe that the performative nature of technology necessitates significant interdisciplinary relationships between scholars in the humanities and in STEM fields. I use the word "necessitates" because, beyond questions of expertise and experience, it is

imperative that the critical questions prevalent in both fields become common points of inquiry for future scholarship. My focus on gender presents a point for such inquiry. As the recognition of gender disparity and masculine "bro" culture become better understood as toxic and limiting, it becomes less apparent how technology itself serves as a point by which gender proliferates. Studies of rhetoric and gender cannot stand on the outside of computer science and engage in such important critiques without also involving experts in the field and in academia to provide insight into how that critique can serve as an agent of change. I am not calling for negotiation of some sort of professional turf, but an honest and, as far as I see, more substantial approach to the question of gender and technology as it is constructed in professional and academic contexts. We must have a goal, surely. We must critique and confront particular forms of dominating or hegemonic masculinity if and as they rise in these particular communities. But we must all enact a collaborative agenda that values such collaboration in and of itself. How we accomplish this is beyond the purview of this argument.

Pedagogy

The classroom is perhaps the most concrete space where technology, writing, rhetoric, and cultural questions of gender overlap. Digital media and connectivity, not to mention multimodal composition and writing studies, have become a staple of pedagogical theory and practice. Rather than reiterate scholarship relating to such a claim, it suffices to say that one of the major questions in writing pedagogy is how digital technology fits into the classroom, more specifically how the emerging body of scholarship on rhetorical code studies fits into these classrooms.

Consider *Slash Goggles* as an exercise. It already presents us with an explicit lesson in how technology can overlap with questions of computation. The lessons that it would provide for approaches, such as procedural rhetoric, are apparent, as are its implications for critical theories of gender. It could also serve as a general introduction into such theories in a course oriented towards digital media and gender, as it asks any number of questions that students could dig into and expand on over the course of an entire term. With a little preparatory work, teachers in gender studies, rhetoric, or composition/writing studies could find any number of approaches to the text. But what about *actual* programming classes? There are some debates about the role of computer code in the writing classroom, but what of actual courses that teach actual, production-oriented coding through large frameworks or on fluid, online platforms? If scholars in the humanities are to take seriously the fact that technology itself is expressive and political, it does not stand to reason that using that technology to teach tradition rhetorical models of communication (as one example) will present equal opportunity to explore the political implications of that technology without spending time investigating the mechanisms of that technology. Such a task would require a more technical vocabulary and background than typically presented in such cases.

I suggest two complementary approaches. The first is a better understanding of collaborative teaching across STEM and humanities disciplines in STEM classrooms that foreground critical and cultural topics like gender as integral to technical fields. Rather than suggesting a generic co-teaching situation, I offer that the integral work of critical gender and rhetorical studies would be to create initiatives geared towards supporting truly interdisciplinary pedagogies geared towards shaping programmatic curriculum and

requirements from the ground up. Rather than efforts to integrate marginal courses that address issues of discourse and gender into existing curriculum in computer science, I argue for critical computational pedagogies focused on addressing the histories and politics of computer programming that are actually grounded in those fields. These kinds of pedagogies could produce engaging and productive coursework in the Humanities and in STEM fields like computer programming so long as they are informed by scholarship and research from both sides. They would also raise significant, shared values between the two disciplines and open up space for conversation and debate that is not argued from across different disciplinary concerns.

For such a curriculum to take shape, my two previous points must come into play. Without serious efforts into collaborative and interdisciplinary research that promotes collaborative, interdisciplinary scholarship, I do not believe that such a pedagogy will present itself. I believe this to be the case because few, if any, rhetorical scholars understand the fundamentals of computer science to such a degree that they could feasibly teach, for example, a critical examination of programming techniques for young professionals preparing to enter the field. This is not a slight against such scholars, because that is not the discipline they study. We wouldn't realistically expect a computer scientist to effectively teach surveys in literature or prepare young graduates to instruct writing courses either. As exemplified in my argument, there are interesting crossroads where critical theories of language and gender intersect with computational culture. It would only increase the potential for real work with significant impact when we start to work together to produce a shared discourse with common goals.

I believe that many of these concerns carry an element of formality that places questions of gender in technology in the background. The fact remains that, in many technical jobs that

involve programming or engineering, women are the vast minority. This fact is itself a problem, but also, I believe, a symptom of the distressing history of gender and digital technology. The representational problem, I contend, is a symptom of (at least in part) structural problems that span styles of collaboration and education. When I slide into talk of collaboration and teaching and rhetoric, what I seek to do is foreground the structural issues that I see unfold in my analysis that are not clear, and often go unnoticed. In calling for more integrated forms of scholarly collaboration and teaching, I am hoping for a form of scholarship to emerge that sees computation and computer programming as critical and cultural areas of study, and that fully calls upon and integrates the knowledge of that field. In writing this dissertation I am convinced that the limitation I faced here, one that will shape my future work, are that I approached it as a solitary author. This is writ large across multiple fields, and I hope that the arguments staged here will inform future work that emphasizes such collaboration as an end goal that creates the conditions of change

Bibliography

Anderson, Nate and Cyrus Farivar. 2013. “How the Feds Took Down the Dread Pirate Roberts.” *Ars Technica.* http://arstechnica.com/tech-policy/2013/10/how-the-feds-took-down-the-dread-pirate-roberts.

Ahmed, Sarah. 2006. “Orientations: Towards a Queer Phenomenology." *GLQ* 12 (4): 543-574.

Barad, Karen. 2003. “Poshumanist Performativity: Toward an Understanding of How Matter

Comes to Matter.” *Signs: Journal of Women in Culture and Society* 28 (3): 801-831.

Beck, Estee. 2015. “The Invisible Digital Identity: Assemblages in Digital Networks.” *Computers and Composition* 35 (1): 125-140.

Blas, Zach. 2008. *transCoder*. *Queer Technologies*. http://www.zachblas.info/projects/queer-technologies.

Blas, Zach and Cardenas, M. 2013. "Imaginary Computational Systems: Queer Technologies and Transreal Aesthetics." *AI and Society* 28 (4): 559-566.

Bogost, Ian. 2007. *Persuasive Games: The Expressive Power of Videogames*. Cambridge, MA: The MIT Press.

Braidotti, Rosi. 2011. *Nomadic Subjects: Embodiment and Sexual Difference in Contemporary Feminist Theory.* New York: Columbia University Press.

---. 2013. *The Posthuman.* Cambridge, UK: Polity Press.

Brito, Jerry, and Andrea Castillo. 2013. “Bitcoin: A Primer for Policy Makers.” *The Mercatus Center*. http://mercatus.org/sites/default/files/Brito_BitcoinPrimer.pdf.

Brock, Kevin. 2012. “One Hundred Thousand Billion Processes: Oulipian Computation and the Composition of Digital Cybertexts." *Technoculture: An Online Journal of Technology and Society* (2). https://tcjournal.org/drupal/vol2/brock.

Brooke, Collin. 2009. *Lingua Fracta: Toward a Rhetoric of New Media.* Cresskill, NJ: Hampton Press.

Brown Jr., James. 2015. *Ethical Programs: Hospitality and the Rhetoric of Software*. Ann Arbor, MI: University of Michigan Press.

Butler, Judith. 1993. *Bodies That Matter: On the Discursive Limits of Sex.* New York: Routledge.

—. 1997. *Excitable Speech: A Politics of the Performative*. New York, NY: Routledge.

—.1990. *Gender Trouble: Feminism and the Subversion of Identity*. New York, NY: Routledge.

—. 2010. “Performative Agency." *Journal of Cultural Economy* 3 (2): 147-161.

Campbell, Karlyn Kohrs. 2005. “Agency: Promiscuous and Promethean.” *Communication and Critical/Cultural Studies* 2 (1): 1-19.

Chang, Edmond. 2015. “Love is in the Air: Queer (Im)possibility and Straightwashing in *Frontierville* and *World of Warcraft.*” *QED: A Journal of GLBTQ Worldmaking* 2 (2): 6-31.

Chun, Wendy. 2011. *Programmed Visions: Software and Memory*. Cambridge, MA: The MIT Press.

—-. 2004. "On Software, or the Persistence of Visual Knowledge." *Grey Room* 18 (Winter):

26-51.

Cohen, Bram. 2012. “The BitTorrent Protocol Specification." BitTorrent.org. http:// www.bittorrent.org/beps.

Cooper, Marilyn M. 2011. “Rhetorical Agency as Emergent and Enacted.” *College Composition and Communication* 62 (3): 420-449.

Cummings, Robert E. 2006. “Coding with Power: Towards a Rhetoric of Computer Coding and Composition." *Computers and Composition* 23: 430-443.

Cushing, Ellen. 2013. "Amazon Mechanical Turk: The Digital Sweatshop." UTNE (February).

Davis, Erin C. 2008. “Situating 'Fluidity': (Trans)Gender Identification and the Regulation of Gender Diversity." *Gay and Lesbian Quarterly* 15 (1): 97-130.

Deleuze, Gilles. 1992. “Postscript on the Societies of Control." October 59 (Winter): 1-8.

Ensmenger, Nathan. 2010. “Making Programming Masculine.” In *Gender Codes: Why Women are Leaving Computing.* Edited by Thomas J. Misa. Hoboken. NJ: Wiley Press, 2010.

Gaboury, Jacob. 2010. "Interview with Zach Blas." *Rhizome* (August). https://rhizome.org/ editorial/2010/aug/18/interview-with-zach-blas.

Galloway, Alexander. 2004. *Protocol*. Cambridge, MA: The MIT Press.

Galloway, Alexander, and Eugene Thacker. 2004. “Protocol, Control, and Networks.” Grey Room 17: 6-29.

—. 2007. *The Exploit*: *A Theory of Networks.* Minneapolis, MN: University of Minnesota Press.

Geisler, Cheryl. 2004. “How Ought We to Understand the Concept of Rhetorical Agency? Report from the ARS.” *Rhetoric Society Quarterly* 34 (3): 9-17. Gerdes, Kendall. 2008. “Performativity." *TSQ* 1 (1): 148-150.

Gill-Peterson, Julian. 2014. “The Technical Capacity of the Body: Assembling Race, Technology, and Transgender.” *Transgender Studies Quarterly* 1 (3): 402-418.

Greene, Ronald Walter. 2004. “Rhetoric and Capitalism: Rhetorical Agency as Communicative Labor.” *Philosophy and Rhetoric* 37 (3): 188-206.

Halberstam, J. Jack. 2011. *The Queer Art of Failure.* Durham, NC: Duke University Press.

Haraway, Donna J. 1991. “A Cyborg Manifesto.” *Simians, Cyborgs, and Women: The Reinvention of Nature.* New York: Routledge.

Hauser, Gerard A. 2004. “Editor’s Introduction.” *Philosophy and Rhetoric* 37 (3): 181-187. Hawisher, Gail E., Cynthia Selfe, Gorjana Kisa, and Shafinaz Ahmed. 2010. “Globalism and Multimodality in a Digitized World: Computers and Composition Studies.” *Pedagogy* 10 (1): 55-68.

Hayles, N. Katherine. 2005. *My Mother was a Computer: Digital Subjects and Literary Texts*

Chicago, IL: University of Chicago Press.

Hawk, Byron. 2004. “Toward a Rhetoric of Network (Media) Culture: Notes on Polarities and Potentiality.” *JAC* 24 (4): 831-850.

Jones, Mark Peter. 1996. “Posthuman Agency: Between Theoretical Traditions.” *Sociological Theory* 14 (3): 290-309.

Kember, Sarah. 2003. *Cyberfeminism and Artificial Life*. New York, NY: Routledge.

Kitchen, Robert, and Martin Dodge. 2011. *Code/Space.* Cambridge, MA: The MIT Press.

Kittler, Friedrich. 1990. *Discourse Networks 1800/1900*. Stanford, CA: Stanford University Press.

Kittur, Aniket. 2010. “Crowdsourcing, Collaboration, and Creativity." XRDS 17 (2).

Latour, Bruno. 1993. *We Have Never Been Modern.* Cambridge, MA: Harvard University Press. Leff, Michael. 2012. “Tradition and Agency in Humanistic Rhetoric.” *Philosophy and Rhetoric* 45 (2): 213-226.

Leff, Michael, and Andrea A. Lunsford. (2004) “Afterwords: A Dialogue.” *Rhetoric Society*

Quarterly 34 (3): 55-67.

Losh, Elizabeth M. 2010. *Virtualpolitik: An Electronic History of Government Media-Making in a Time of War, Scandal, Disaster, Miscommunication, and Mistakes.* Cambridge, MA: The MIT Press.

Lundberg, Christian and Joshua Gunn. 2005. “’Ouija Board, Are There Any Communications?’ Agency, Ontotheology, and the Death of the Humanist Subject, or, Continuing the ARS Conversation.” *Rhetoric Society Quarterly* 35 (5): 83-105.

Mackenzie, Adrian. 2005. "The Performativity of Code: Software and Cultures of Circulation." *Theory, Culture, and Society* 22 (1): 71-92.

Mailloux, Steven. 2012. “Humanist Controversies: The Rhetorical Humanism of Ernesto Grassi and Michael Leff.” *Philosophy and Rhetoric* 45 (2): 134-147.

Marino, Mark C. 2004. “Critical Code Studies.” *Electronic Book Review* (April). http://www.electronicbookreview.com/thread/electropoetics/codology.

---. 2012. “Of Sex, Cylons, and Worms: A Critical Code Study of Heteronormativity." *Leonardo Electronic Almanac* 17 (2): 184-20.

Marvit, Moshe Z. 2014. “How Crowdworkers Became Ghosts in the Digital Machine." *The Nation* (February). http://www.thenation.com/article/how-crowdworkers-becameghosts-digital-machine.

McLuhan, Marshall. 1964. *Understanding Media: The Extensions of Man.* New York: McGraw-Hill.

Miller, Carolyn. 2007. “What Can Automation Tell us About Agency?” *Rhetoric Society Quarterly* 37 (2): 137-157.

Muñoz, José Esteban. 1999. *Disidentifications: Queers of Color and the Performance of Politics*. Minneapolis, MN: University of Minnesota Press.

Nicholas, Lucy. 2014. *Queer Post-Gender Ethics: The Shape of Selves to Come.* New York:

Palgrave Macmillan.

O’Niell, Patrick Howell. 2015. "As Netflix’s Internet Traffic Soars, BitTorrent’s Continues to Plunge." The Daily Dot. *dailydot.com*. http://www.dailydot.com/debug/netflix-killing-bittorrent.

Penwar, et al. 2004. *TCP/IP Essentials: A Lab-Based Approach*. New York, NY: Cambridge University Press.

Porter, James. 2009. "Recovering Delivery for Digital Rhetoric." *Computers and Composition* 26 (4): 207-224.

Trimbur, John. 2000. “Composition and the Circulation of Writing." *College Composition and Communication* 52 (2): 188-219.

Rand, Erin J. 2014. *Reclaiming Queer: Activist & Academic Rhetorics of Resistance*. Tuscaloosa: University of Alabama Press.

Rice, Jenny. 2005. “Unframaing Models of Public Distribution: From Rhetorical Situation to

Rhetorical Ecologies.” *Rhetoric Society Quarterly* 35 (4): 5-24.

Ruberg, Bonnie. 2015. ¨No Fun: The Queer Potential of Video Games that Annoy, Anger,

Disappoint, Sadden, and Hurt." *QED* 2 (2): 108-124

Russo, Julie L. 2008. “Visual Informatics – The *Slash Goggles* Algorithm." Last modified April 10, 2008. http://thearchive2.livejournal.com/1465.html.

Sedgwick, Eve K. 1990. *The Epistemology of the Closet.* Berkeley: University of California Press.

Stevens, W. Richard. 1994. *TCP/IP Illustrated, Volume 1*. New York, NY: Addison-Wesley.

Stormer, Nathan. 2004. “Articulation: A Working Paper on Rhetoric and *Taxis.*” *Quarterly Journal of Speech* 90 (3): 257-284.

Terranova, Tiziana. 2004. *Network Cultures*. Ann Arbor MI: Pluto Press.

Trabex, Winter. 2014 “Bitcoin is a Currency of Equality.” *The Art of Not Being Governed*.

notbeinggoverned.com. http://www.notbeinggoverned.com/bitcoin-currency-equality.

Wardrip-Fruin, Noah. 2009. *Expressive Processing: Digital Fictions, Computer Games, and*

Software Studies. Cambridge, MA: The MIT Press.

Watts, William L. 2013. “Bitcoin Hits Record $1,242 As It Nears Value of Ounce of Gold.” Market Watch. http://blogs.marketwatch.com/thetell/2013/11/29/bitcoin-hits-record-1242-as-it-nears-value-of-ounce-of-gold.

"Who We Are." HarassMap. harassmap.org. http://harassmap.org/en/who-we-are.

Wysocki, Anne, Johndan Johnson-Eilola, Cynthia L. Selfe, and Geoffrey Sirc, eds. 2003. *Writing New Media: Theory and Application for Expanding the Teaching of Composition.* Salt Lake City, UT: University of Utah Press.

Young, Chelsea. 2014 "HarassMap: Using Crowdsourced Data to Map Sexual Harassment in Egypt." *Technology Innovation Management Review* (March). https://timreview.ca/ article/770.

Zappen, James. 2005. “Digital Rhetoric: Toward an Integrated Theory.” *Technical Communication Quarterly* 14 (3): 319-325.